Groupe de travail Réseau
Request for Comments : 3253
Catégorie : En cours de normalisation

Traduction Claude Brière de L'Isle

G. Clemm, Rational Software
J. Amsden & T. Ellison, IBM
C. Kaler, Microsoft
J. Whitehead, U.C. Santa Cruz
mars 2002

Extensions de versions à WebDAV (Protocole de collecte ordonnée des auteurs et des versions distribuée sur la Toile)

Statut de ce mémoire
Le présent document spécifie un protocole Internet en cours de normalisation pour la communauté de l'Internet, et appelle à des discussions et des suggestions pour son amélioration. Prière de se reporter à l'édition actuelle du STD 1 "Normes des protocoles officiels de l'Internet" pour connaître l'état de normalisation et le statut de ce protocole. La distribution du présent mémoire n'est soumise à aucune restriction.

Résumé
Le présent document spécifie un ensemble de méthodes, en-têtes, et types de ressources qui définissent les extensions au protocole HTTP/1.1 des versions du protocole de collecte ordonnée des auteurs et des versions distribuée sur la Toile (WebDAV, *Web Distributed Authoring and Versioning*). La gestion de versions de WebDAV va minimiser la complexité des clients qui sont capables d'interopérer avec divers gestionnaires de répertoires de versions, de faciliter un large déploiement des applications capables d'utiliser les services de gestion de versions de WebDAV. La gestion de versions de WebDAV inclut la gestion automatique des versions pour les clients incapables de gérer les versions, la gestion de l'historique des versions, la gestion de l'espace de travail, la gestion de base, la gestion d'activité, et la gestion d'espace de nom d'URL.

Table des Matières

1. Introduction

Le présent document spécifie un ensemble de méthodes, en-têtes, et propriétés qui définissent les extensions de collecte de versions de WebDAV *(Web Distributed Authoring and Versioning)* au protocole HTTP/1.1. La collecte des versions est concernée par la recherche et l'accès à l'historique des états importants d'une ressource de la Toile, comme une page autonome de la Toile. Les avantages de la collecte de versions dans le contexte de la Toile mondiale sont :

- Qu'une ressource a une histoire explicite et une identité persistante à travers les divers états qu'elle a eu durant le cours de cette histoire. Cela permet de naviguer à travers le passé et les différentes versions d'une ressource. L'histoire des modifications et de la succession des auteurs d'une ressource est fréquemment une information critique par elle-même.
- Les états de ressource (les versions) reçoivent des noms stables qui peuvent prendre en charge des liens mémorisés en externe pour la prise en charge d'un serveur d'annotations et de liens. Les serveurs d'annotation aussi bien que de liens ont fréquemment besoin de mémoriser des références stables à des portions de ressources qui ne sont pas sous leur contrôle direct. En fournissant des états stables de ressources, les systèmes de contrôle de version permettent non seulement des pointeurs stables sur ces ressources, mais aussi des méthodes bien définies de détermination des relations de ces états d'une ressource.

La collecte de versions WebDAV définit les deux fonctions de base et évoluée de collecte de versions.

La collecte de version de base permet aux utilisateurs :

- d'établir un contrôle de version sur une ressource
- de déterminer si une ressource est à version contrôlée
- de déterminer si une mise à jour de ressource sera automatiquement capturée comme nouvelle version
- de créer un accès distinct aux versions d'une ressource.

La collecte évoluée de versions donne des fonctions supplémentaires pour un développement en parallèle et la gestion de configuration d'ensembles de ressources de la Toile.

Le présent document va d'abord définir la sémantique des propriétés et des méthode pour les caractéristiques de la collecte de version de base, puis définir la sémantique des propriétés et méthodes additionnelles pour les caractéristiques de collecte de version évoluée. Une mise en œuvre qui serait seulement intéressée par la collecte de versions de base peut sauter les sections consacrées à la collecte de version évoluée (les sections 10 à 14).

1.1 Relations avec WebDAV

Pour maximiser l'interopérabilité et l'utilisation des fonctions de protocole existantes, la prise en charge de la collecte de version est conçue comme une extension au protocole WebDAV [RFC2518], qui lui-même est une extension au protocole HTTP [RFC2616]. Tous tris et post-conditions de méthode définis dans les RFC2518 et RFC2616 continuent de s'appliquer, pour s'assurer que les clients qui ignorent la collecte de versions peuvent interopérer avec succès avec les serveurs de collecte de version. Bien que les extensions de collecte de version soient conçues pour être orthogonales à la plupart des aspects des protocoles WebDAV et HTTP, une précision à la RFC2518 est nécessaire pour rendre effective l'interopérabilité de la collecte de versions. Cette précision est décrite au paragraphe 1.7.

1.2 Conventions de notation

Dans le présent document, les mots clés "DOIT", "NE DOIT PAS", "EXIGE", "DEVRA", "NE DEVRA PAS", "DEVRAIT", "NE DEVRAIT PAS", "RECOMMANDÉ", "PEUT", et "FACULTATIF" sont à interpréter comme décrit dans le BCP 14, [RFC2119].

Le terme "protégé" est placé entre parenthèses à la suite de la définition d'une propriété protégée (voir le paragraphe 1.4.2).

Le terme "calculé" est placé entre parenthèses à la suite de la définition d'une propriété calculée (voir le paragraphe 1.4.3).

Lorsque un type d'élément XML dans l'espace de noms "DAV:" est référencé dans le présent document en dehors du contexte d'un fragment XML, la chaîne "DAV:" sera mise en préfixe au type d'élément.

Lorsque une méthode est définie dans le présent document, une liste de préconditions et de postconditions sera définie pour

cette méthode. Si la sémantique d'une méthode existante est étendue, une liste de préconditions et postconditions additionnelles sera définie. Une précondition ou postcondition est précédée d'un type d'élément XML entre parenthèses qui identifie cette précondition ou postcondition (voir le paragraphe 1.6).

1.3 Termes

Le présent document utilise les termes définis dans la RFC2616, dans la RFC2518, et dans la présente section. Le paragraphe 2.2 définit la sémantique du modèle de collecte de versions sous-jacent à cette terminologie.

Contrôle de version, enregistré, désenregistré
"Contrôle de version" est un ensemble de contraintes sur la façon dont une ressource peut être mise à jour. Une ressource sous contrôle de version est dans un état soit "enregistré" soit "désenregistré", et les contraintes du contrôle de version ne s'appliquent que lorsque la ressource est dans l'état enregistré.

Ressource versionisable
Une "ressource versionisable" est une ressource qui peut être mise sous contrôle de version.

Ressource à version contrôlée
Lorsque une ressource versionisable est mise sous contrôle de version, elle devient une "ressource à version contrôlée". Une ressource à version contrôlée peut être "désenregistrée" pour permettre des modification de son contenu ou de propriétés mortes par des méthodes standard HTTP et WebDAV.

Ressource désenregistrée
Une "ressource désenregistrée" est une ressource sous contrôle de version qui est dans l'état désenregistré.

Version de ressource
Une "version de ressource", ou simplement "version", est une ressource qui contient une copie d'un état particulier (contenu et propriétés mortes) d'une ressource à version contrôlée. Une version est créée par "l'enregistrement" d'une ressource désenregistrée. Le serveur alloue un nouvel URL distinct pour chaque nouvelle version, et cet URL ne va jamais être utilisé pour identifier une ressource autre que cette version. Le contenu et les propriétés mortes d'une version ne changent jamais.

Ressource d'historique de version
Une "ressource d'historique de version", ou simplement "historique de version", est une ressource qui contient toutes les versions d'une ressource à version contrôlée particulière.

Nom de version
Un "nom de version" est une chaîne choisie par le serveur pour distinguer une version d'un historique de version des autres versions de cet historique de version. Les versions provenant de différents historiques de version peuvent avoir le même nom de version.

Prédécesseur, successeur, ancêtre, descendant
Lorsque une ressource à version contrôlée est désenregistrée puis ensuite enregistrée, la version qui a été désenregistrée devient un "prédécesseur" de la version créée par l'enregistrement. Un client peut spécifier plusieurs prédécesseurs pour une nouvelle version si la nouvelle version est logiquement une fusion de ces prédécesseurs. Lorsque une version est connectée à une autre version en traversant une ou plusieurs relations de prédécesseur, elle est appelée un "ancêtre" de cette version. L'inverse de la relation de prédécesseur et d'ancêtre est la relation de "successeur" et de "descendant". Donc, si X est un prédécesseur de Y, alors Y est un successeur de X, et si X est un ancêtre de Y, alors Y est un descendant de X.

Version racine de ressource
La "version racine de ressource", ou simplement "version racine", est la version qui dans un historique de versions est un ancêtre pour toutes les autres versions.

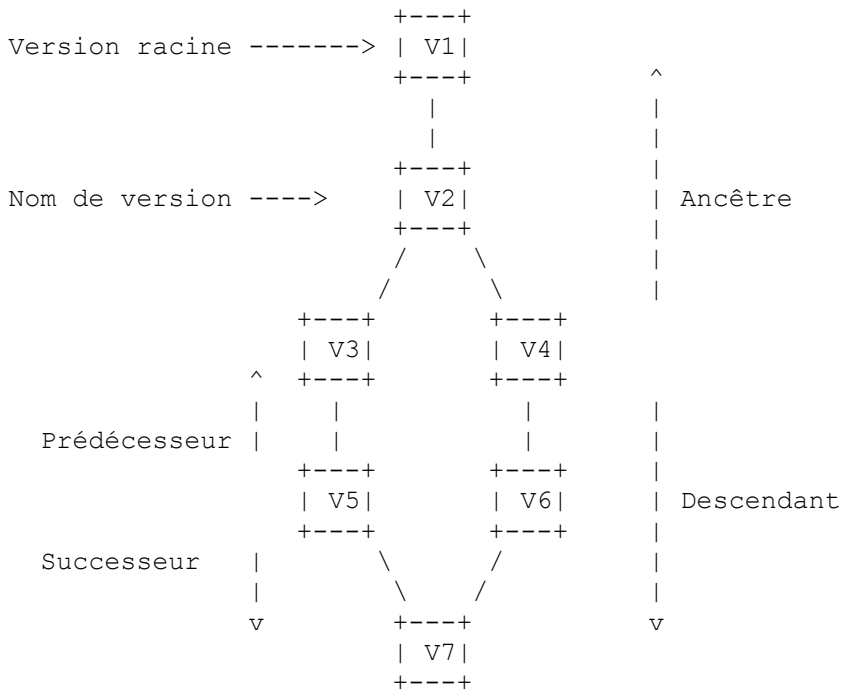
Ressource d'espace de travail
Une "ressource d'espace de travail", ou simplement "espace de travail", est une collection qui contient au plus une ressource à version contrôlée d'un certain historique de version (voir la Section 6).

Ressource de travail
Une "ressource de travail" est une ressource enregistrée créée par le serveur à un URL défini par lui lorsque une version (au lieu d'une ressource à version contrôlée) est désenregistrée. À la différence d'une ressource à version contrôlée désenregistrée, une ressource de travail est supprimée lorsque elle est désenregistrée.

Fourche, fusion
Lorsque un second successeur est ajouté à une version, cela crée une "fourche" dans l'historique de version. Lorsque une version est créée avec plusieurs prédécesseurs, cela crée une "fusion" dans l'historique de version. Un serveur peut restreindre l'historique de version à être linéaire (sans fourche ni fusion) mais un client de collecte de version interopérable devrait être prêt à traiter aussi bien des fourches que des fusions dans l'historique de version.

Le diagramme qui suit illustre plusieurs des définitions suivantes. Chaque boîte représente une version et chaque ligne entre deux boîtes représente une relation de prédécesseur/successeur. Par exemple, il montre que V3 est un prédécesseur de V5, que V7 est un successeur de V5, que V1 est un ancêtre de V4, et que V7 est un descendant de V4. Il montre aussi qu'il y a une fourche à la version V2 et une fusion à la version V7.

Historique de foo.html



Étiquette
Une "étiquette" est un nom qui peut être utilisé pour choisir une version dans un historique de version. Une étiquette peut être allouée par un client ou par le serveur. La même étiquette peut être utilisée dans différents historiques de version.

1.4 Valeurs de propriété

1.4.1 Valeur de propriété initiale

Sauf si la valeur initiale d'une propriété d'un certain type est définie par le présent document, la valeur initiale d'une propriété de ce type dépend de la mise en œuvre.

1.4.2 Valeur de propriété protégée

Lorsque une propriété d'une sorte spécifique de ressource est "protégée", la valeur de la propriété ne peut pas être mise à jour sur cette sorte de ressource sauf par une méthode explicitement définie comme mettant à jour cette propriété spécifique. En particulier, une propriété protégée ne peut pas être mise à jour par une demande PROPPATCH. Noter que une certaine propriété peut être protégée sur une sorte de ressource, mais pas sur une autre.

1.4.3 Valeur de propriété calculée

Lorsqu'une propriété est "calculée", sa valeur est définie en termes de calcul fondé sur le contenu et les autres propriétés de cette ressource, ou même de quelque autre ressource. Lorsque la sémantique d'une méthode est définie dans le présent document, l'effet de cette méthode sur les propriétés non calculées sera spécifié ; l'effet de cette méthode sur les propriétés calculées ne sera pas spécifié, mais peut être déduit du calcul défini pour ces propriétés. Une propriété calculée est toujours une propriété protégée.

1.4.4 Valeur de propriété booléenne

Certaines propriétés prennent une valeur booléenne de "faux" ou "vrai".

1.4.5 Valeur de propriété DAV:href

L'élément XML DAV:href est défini dans la RFC 2518, au paragraphe 12.3.

1.5 Éléments XML d'espace de nom DAV dans les corps de demande et de réponse

Bien que les corps de demande et réponse WebDAV puissent être étendus par des éléments XML arbitraires, qui peuvent être ignorés par le receveur du message, un élément XML dans l'espace de noms DAV NE DOIT PAS être utilisé dans le corps de demande ou réponse d'une méthode de collecte de versions sauf si cet élément XML est explicitement défini dans une RFC de l'IETF.

1.6 Préconditions et postconditions de méthode

Une "précondition" d'une méthode décrit l'état du serveur qui doit être vrai pour cette méthode afin d'être effectuée. Une "postcondition" d'une méthode décrit l'état du serveur qui doit être vrai après que cette méthode a été achevée. Si une précondition ou postcondition de méthode pour une demande n'est pas satisfaite, l'état de réponse à la demande DOIT être soit 403 (Interdit) si la demande ne devrait pas être répétée parce que elle va toujours échouer, ou 409 (Conflit) si on s'attend à ce que l'utilisateur puisse être capable de résoudre le conflit et resoumettre la demande.

Afin de permettre un meilleur traitement par le client des réponses 403 et 409, un type d'élément XML distinct est associé à chaque précondition et postcondition de méthode d'une demande. Lorsque une précondition particulière n'est pas satisfaite ou lorsque une postcondition particulière ne peut pas être réalisée, l'élément XML approprié DOIT être retourné comme enfant d'un élément DAV:error de niveau supérieur dans le corps de la réponse, sauf si la demande l'a négocié autrement. Dans une réponse 207 multi-état, l'élément d'erreur DAV: apparaîtrait dans l'élément approprié DAV:responsedescription.

1.6.1 Exemple – demande CHECKOUT avec réponse DAV:must-be-checked-in

```
>>REQUEST
  CHECKOUT /foo.html HTTP/1.1
  Host: www.webdav.org

>>RESPONSE
  HTTP/1.1 409 Conflict
  Content-Type: text/xml; charset="utf-8"
  Content-Length: xxxx
  <?xml version="1.0" encoding="utf-8" ?>
  <D:error xmlns:D="DAV:">
    <D:must-be-checked-in/>
  </D:error>
```

Dans cet exemple, la demande de CHECKOUT /foo.html échoue parce que /foo.html n'est pas enregistré.

1.7 Éclaircissements sur la sémantique de COPY avec Overwrite:T

Le paragraphe 8.8.4 de la RFC2518 déclare :

"Si il existe une ressource à la destination et si l'en-tête Overwrite est "T", alors avant d'effectuer la copie, le serveur DOIT effectuer un DELETE avec "Depth:infinity" sur la ressource de destination."

L'objet de cette phrase est d'assurer que à la suite d'une COPY, toutes les ressources de destination ont les mêmes contenu et propriétés mortes que les ressources correspondantes identifiées par l'URL de demande (où une ressource avec un certain nom relatif à l'URL de destination "correspond" à une ressource avec le même nom relatif à l'URL de demande). Si au moment de la demande, il y a déjà une ressource à la destination qui a le même type de ressource que la ressource correspondante à l'URL de demande, cette ressource NE DOIT PAS être supprimée, mais DOIT être mise à jour pour avoir le contenu et les propriétés mortes de son membre correspondant. Si un client souhaite que toutes les ressources à la destination soient supprimées avant la COPY, il DOIT explicitement produire une demande DELETE.

La différence entre mettre à jour une ressource et remplacer une ressource par une nouvelle ressource est particulièrement importante lorsque l'historique de ressource est entretenu (le premier ajoute à un historique existant, tandis que le second crée un nouvel historique). De plus, les contraintes de verrouillage et de contrôle d'accès peuvent permettre de mettre à jour une ressource, mais ne pas permettre de la supprimer et d'en créer une autre à la place.

Noter que cette précision ne s'applique pas à une demande MOVE. Une demande MOVE avec Overwrite:T DOIT effectuer le DELETE avec "Depth:infinity" sur la ressource de destination avant d'effectuer le MOVE.

1.8 Méthodes de gestion de version avec verrouillage en écriture

Si une ressource verrouillée en écriture a une propriété non calculée définie par le présent document, la valeur de propriété NE DOIT PAS être changée par une demande sauf si le jeton de verrou approprié est inclus dans la demande. Comme chaque méthode introduite dans le présent document autre que REPORT modifie au moins une propriété définie par ce document, chaque méthode de gestion de version autre que REPORT est affectée par un verrouillage en écriture. En particulier, la méthode DOIT échouer avec un état 423 (Verrouillé) si la ressource est verrouillée en écriture et si le jeton approprié n'est pas spécifié dans un en-tête de demande If.

2. Caractéristique de base de la gestion de version

Chaque caractéristique de base de gestion de version définit des extensions aux méthodes HTTP et WebDAV existantes, ainsi que de nouveaux types de ressources, de propriétés vivantes, et de méthodes.

2.1 Paquetages de base de la gestion de version

Un serveur PEUT accepter toute combinaison de caractéristiques de collecte de versions. Cependant, afin de minimiser la complexité d'un client de base de collecte de version WebDAV, un serveur WebDAV de base de collecte de versions DEVRAIT prendre en charge un des trois "paquetages" (ensembles de caractéristiques) suivants :
- paquetage de cœur de collecte de version : contrôle de version ;
- paquetage de base d'espace de travail de serveur : contrôle de version, espace de travail, historique de version, désenregistré ;
- paquetage de base d'espace de travail client : contrôle de version, ressource de travail, mise à jour, étiquette.

Le paquetage de cœur de collecte de version prend en charge la collecte de version linéaire par les clients aussi bien capables de collecte de version que par ceux qui ne le sont pas. Un client capable de collecte de version peut utiliser les rapports et les propriétés pour accéder aux versions précédentes d'une ressource à version contrôlée.

Les paquetages de base d'espace de travail prennent en charge le développement en parallèle des ressources à version contrôlée. Chaque client a sa propre configuration de la ressource à version contrôlée partagée, et peut faire des changements à sa configuration sans perturber celle d'un autre client.

Dans le paquetage de base d'espace de travail de serveur, tout état persistent est conservé sur le serveur. Chaque client a sa propre ressource d'espace de travail qui est allouée sur le serveur, et chaque espace de travail identifie une configuration des ressources à version contrôlée partagées. Chaque client fait des changements à son espace de travail, et peut transférer les changements lorsque c'est approprié, d'un espace de travail à un autre. Le paquetage d'espace de travail de serveur est approprié pour les clients sans état local persistent, ou pour les clients qui souhaitent exposer leurs configurations de travail aux autres clients.

Dans le paquetage de base d'espace de travail de client, chaque client conserve dans une mémorisation locale persistante l'état de sa configuration de la ressource à version contrôlée partagée. Lorsque un client est prêt à rendre ses changements visibles aux autres clients, il alloue un ensemble de "ressources de travail" sur le serveur, met à jour le contenu et les propriétés mortes de ces ressources de travail, et utilise ensuite l'ensemble de ressources de travail pour mettre à jour les ressources à version contrôlée. Les ressources de travail sont utilisées, au lieu de mettre directement à jour les ressources à version contrôlée, afin que des ensembles de mises à jour cohérents puissent être préparés en parallèle par plusieurs clients. Aussi, une ressource de travail permet à un client de préparer une seule mise à jour qui exige plusieurs demandes de serveur (par exemple, de mettre à jour à la fois le contenu et les propriétés mortes d'une ressource exige à la fois un PUT et un PROPPATCH). Le paquetage d'espace de travail de client simplifie la mise en œuvre de serveur en exigeant que chaque client entretienne son propre espace de noms, mais cela exige que les clients aient un état local persistent, et ne permet pas

aux clients d'exposer leurs configurations de travail aux autres clients.

Un serveur qui prend en charge les deux paquetages de base d'espace de travail va interopérer avec tous les clients de collecte de version de base.

2.2 Sémantique de base de la gestion de version

2.2.1 Création d'une ressource à version contrôlée

Afin de retracer l'historique du contenu et des propriétés mortes d'une ressource versionisable, un usager peut mettre la ressource sous contrôle de version avec une demande VERSION-CONTROL. Une demande VERSION-CONTROL effectue trois opérations distinctes :

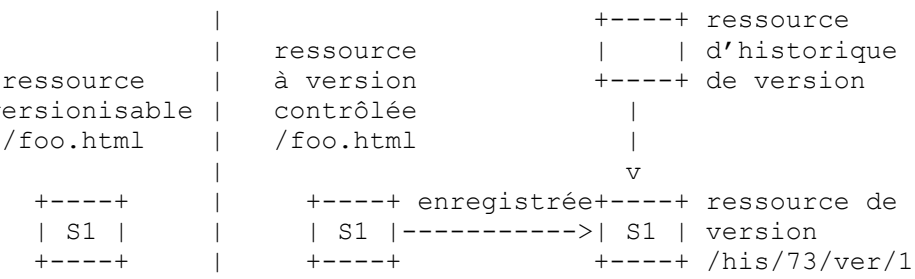
1) Elle crée une nouvelle "ressource d'historique de version". Dans le versionnage de base, une ressource d'historique de version n'a pas d'URL alloué, et n'est donc pas visible dans l'espace d'URL du schéma http. Cependant, lorsque la caractéristique d'historique de version (voir la Section 5) est prise en charge, cela change, et chaque ressource d'historique de version reçoit un nouvel URL distinct et unique défini par le serveur.
2) Elle crée une nouvelle "ressource de version" et l'ajoute à la nouvelle ressource d'historique de version. Le corps et les propriétés mortes de la nouvelle ressource de version sont une copie de celles de la ressource versionisable. Le serveur alloue à la nouvelle ressource de version un nouvel URL distinct et unique.
3) Elle convertit la ressource versionisable en une "ressource à version contrôlée". La ressource à version contrôlée continue d'être identifiée par le même URL qui l'identifiait comme ressource versionisable. Au titre de cette conversion, elle ajoute une propriété DAV:checked-in, dont la valeur contient l'URL de la nouvelle ressource de version.

Noter qu'une ressource versionisable et une ressource à version contrôlée ne sont pas de nouveaux types de ressources (c'est-à-dire qu'elles n'introduisent pas de nouveau DAV:resourcetype), mais elles sont plutôt de tout type de ressource qui accepte les méthodes et les propriétés vives définies pour elles dans le présent document, en plus de toutes les méthodes et propriétés vives impliquées par leur DAV:resourcetype. Par exemple, une collection (dont le DAV:resourcetype est DAV:collection) est une ressource versionisable si elle accepte la méthode VERSION-CONTROL, et est une ressource à version contrôlée si elle accepte les méthodes et les propriétés vives de ressource à version contrôlée.

Dans l'exemple suivant, foo.html est une ressource versionisable qui est mise sous contrôle de version. Après la réussite de la demande VERSION-CONTROL, il y a deux ressources supplémentaires : une nouvelle ressource d'historique de version et une nouvelle ressource de version dans cet historique de version. La ressource versionisable est convertie en une ressource de version contrôlée, dont la propriété DAV:checked-in identifie la nouvelle ressource de version.

Le contenu et les propriétés mortes d'une ressource sont représentés par le symbole qui apparaît à l'intérieur de la boîte pour cette ressource (par exemple, "S1" dans l'exemple qui suit).

===VERSION-CONTROL==>



Donc, alors qu'avant la demande VERSION-CONTROL il y avait seulement une ressource de version non contrôlée, après la VERSION-CONTROL, il y a trois ressources séparées, distinctes, dont chacune contient son propre état et propriétés : la ressource de version contrôlée, la ressource de version, et la ressource d'historique de version. Comme la ressource de version contrôlée et la ressource de version sont séparées, distinctes, lorsque une méthode est appliquée à une ressource de version contrôlée, elle n'est appliquée qu'à cette ressource de version contrôlée, et n'est pas appliquée à la ressource de version qui est actuellement identifiée par la propriété DAV:checked-in de cette ressource de version contrôlée. Bien que le contenu et les propriétés mortes d'une ressource de version contrôlée enregistrée soient obligés d'être les mêmes que ceux de sa version actuelle de DAV:checked-in, ses propriétés vives peuvent différer. Une mise en œuvre peut optimiser la mémorisation en restituant le contenu et les propriétés mortes d'une ressource de version contrôlée enregistrée à partir de sa version actuelle de DAV:checked-in plutôt que de les mémoriser dans la ressource de version contrôlée, mais ceci est juste une optimisation de mise en œuvre.

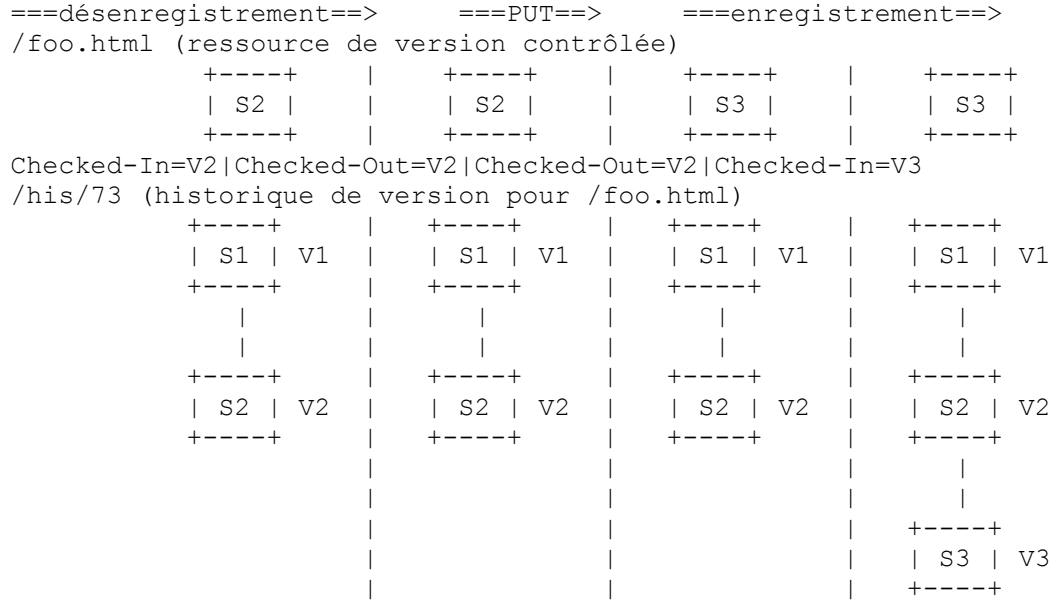
Normalement, une ressource est placée sons contrôle de version par une demande VERSION-CONTROL explicite. Un serveur PEUT placer automatiquement chaque nouvelle ressource versionisable sous contrôle de version. Dans ce cas, l'état résultant sur le serveur DOIT être le même que si le client avait explicitement appliqué une demande VERSION-CONTROL à la ressource versionisable.

2.2.2 Modification d'une ressource à version contrôlée

Afin d'utiliser des méthodes comme PUT et PROPPATCH pour modifier directement le contenu ou les propriétés mortes d'une ressource de version contrôlée, la ressource de version contrôlée doit d'abord être désenregistrée. Lorsque la ressource désenregistrée est enregistrée, une nouvelle version est créée dans l'historique de version de cette ressource de version contrôlée. La version qui a été désenregistrée est mémorisée comme prédécesseur de la nouvelle version.
La propriété DAV:auto-version (voir le paragraphe 3.2.2) d'une ressource de version contrôlée enregistrée détermine comment elle répond à une méthode qui tente de modifier son contenu ou ses propriétés mortes. Les réponses possibles incluent :

- Échec de la demande. La ressource exige une demande explicite CHECKOUT pour qu'elle soit modifiée (voir la Section 4 et le paragraphe 9.2.1).
- Désenregistrer automatiquement la ressource, effectuer la modification, et enregistrer automatiquement la ressource. Cela assure que tout état de la ressource est retracé par le serveur, mais peut résulter en un nombre excessif de versions créées.
- Désenregistrer automatiquement la ressource, effectuer la modification, et ensuite, si la ressource n'est pas verrouillée en écriture, enregistrer automatiquement la ressource. Si la ressource est verrouillée en écriture, elle reste désenregistrée jusqu'à ce que le verrouillage en écriture soit retiré (soit explicitement par une demande UNLOCK ultérieure soit implicitement par une fin de temporisation du verrouillage en écriture). Cela aide le client qui verrouille à éviter la prolifération de versions, tout en permettant quand même à un client qui ne verrouille pas de mettre à jour la ressource.
- Désenregistrer automatiquement la ressource, effectuer la modification, et ensuite laisser la ressource désenregistrée. Si la ressource est verrouillée en écriture, elle sera automatiquement enregistrée lorsque le verrouillage en écriture sera retiré, mais une opération CHECKIN explicite (voir au paragraphe 4.4) est nécessaire pour une ressource non verrouillée en écriture. Cela minimise le nombre de nouvelles versions qui seront créées par un client sans capacité de gestion des versions, mais seulement un client capable de gérer les versions peut créer de nouvelles versions d'une ressource non verrouillée en écriture.
- Échec de la demande sauf si la ressource est verrouillée en écriture. Si elle est verrouillée en écriture, désenregistrer automatiquement la ressource et effectuer la modification. La ressource est automatiquement enregistrée lorsque le verrouillage en écriture est retiré. Cela minimise le nombre de nouvelles versions qui seront créées par un client incapable de gérer les versions, mais ne désenregistre jamais automatiquement une ressource qui ne va pas être ensuite automatiquement enregistrée.

Le diagramme qui suit illustre l'effet du processus d'enregistrement/désenregistrement sur une ressource de version contrôlée et son historique de version. Le symbole à l'intérieur d'une boîte (S1, S2, S3) représente le contenu actuel et les propriétés mortes de la ressource représentée par cette boîte. Le symbole après une boîte (V1, V2, V3) représente l'URL pour cette ressource.



Noter qu'une version capture seulement un sous ensemble défini de l'état d'une ressource. En particulier, une version d'une ressource de base capture son contenu et ses propriétés mortes, mais pas ses propriétés vives.

2.2.3 Rapports

Certaines informations de gestion de version sur une ressource exigent que des paramètres soient spécifiés avec cette demande d'informations. Dans le versionnage de base est incluse la prise en charge exigée d'un mécanisme extensible de rapports, qui inclut une méthode REPORT ainsi qu'une propriété vive pour déterminer quels rapports sont acceptés par une ressource particulière. La méthode REPORT est exigée pour le versionnage, mais elle peut être utilisée dans des extensions WebDAV étrangères au versionnage.

Pour permettre à un client d'interroger les propriétés de toutes les versions dans l'historique de version d'une ressource de version contrôlée spécifiée, le versionnage de base fournit le rapport DAV:version-tree (voir au paragraphe 3.7). Un mécanisme de rapport d'historique de version plus puissant est donné en appliquant le rapport DAV:expand-property (voir au paragraphe 3.8) à une ressource d'historique de version (voir la Section 5).

3. Caractéristiques du contrôle de version

La caractéristique contrôle de version permet la prise en charge de la mise sous contrôle de version d'une ressource, en créant une ressource de version contrôlée et une ressource d'historique de version associées, comme décrit au paragraphe 2.2.1. Un serveur indique qu'il prend en charge la caractéristique de contrôle de version en incluant la chaîne "version-control" comme champ dans l'en-tête DAV de la réponse à une demande OPTIONS. La caractéristique de contrôle de version DOIT être prise en charge si une autre caractéristique de versionnage est prise en charge.

3.1 Propriétés de ressources supplémentaires

La caractéristique contrôle de version introduit les propriétés EXIGÉES suivantes pour toute ressource WebDAV.

3.1.1 DAV:comment

Cette propriété est utilisée pour tracer un bref commentaire sur une ressource, convenable pour la présentation à un usager. Le commentaire DAV:comment d'une version peut être utilisé pour indique pourquoi cette version a été créée.

<!ELEMENT commentaire (#PCDATA)>
valeur de PCDATA : chaîne

3.1.2 DAV:creator-displayname

Cette propriété contient une description du créateur de la ressource, qui convient pour une présentation à un usager. Le DAV:creator-displayname *(affichage du nom du créateur)* d'une version peut être utilisé pour indiquer qui a créé cette version.

<!ELEMENT creator-displayname (#PCDATA)>
valeur de PCDATA : chaîne

3.1.3 DAV:supported-method-set (protégé)

Cette propriété identifie les méthodes qui sont prises en charge par la ressource. Une méthode est prise en charge par une ressource si il y a un état de cette ressource pour lequel une application de cette méthode va réussir à satisfaire toutes les postconditions de cette méthode, y compris toutes postconditions supplémentaires ajoutées par les caractéristiques prises en charge par cette ressource.

<!ELEMENT ensemble-de-méthodes-prises-en-charge (méthode-prise-en-charge*)>
<!ELEMENT méthode-prise-en-charge TOUTE>
<!ATTLIST nom de méthode-prise-en-charge NMTOKEN #EXIGÉ>
valeur de nom : un nom de méthode

3.1.4 DAV:supported-live-property-set (protégé)

Cette propriété identifie les propriétés vives qui sont prises en charge par la ressource. Une propriété vive est prise en

charge par une ressource si cette propriété a la sémantique définie pour cette propriété. La valeur de cette propriété DOIT identifier toutes les propriétés vives définies par le présent document qui sont acceptées par la ressource, et DEVRAIT identifier toutes les propriétés vives qui sont prises en charge par la ressource.

<!ELEMENT ensemble-de-propriétés-vives-prises-en-charge (propriété-vive-prise-en-charge*)>
<!ELEMENT nom de propriété-vive-prise-en-charge>
<!ELEMENT prop TOUTE>
valeur de TOUTE : un type d'élément de propriété

3.1.5 DAV:supported-report-set (protégé)

Cette propriété identifie les rapports qui sont pris en charge par la ressource.

<!ELEMENT ensemble-de-rapports-pris-en-charge (rapport-pris-en-charge*)>
<!ELEMENT rapport de rapport-pris-en-charge>
<!ELEMENT rapport TOUTE>
valeur de TOUTE : un type d'élément de rapport

3.2 Propriétés de ressource à version contrôlée

La caractéristique version-control introduit les propriétés EXIGÉES suivantes pour une ressource de version contrôlée.

3.2.1 DAV:checked-in (protégé)

Cette propriété apparaît sur une ressource de version contrôlée enregistrée, et identifie une version qui a le même contenu et propriétés mortes que la ressource de version contrôlée. Cette propriété est retirée lorsque la ressource est désenregistrée, et ensuite rajoutée (identifiant une nouvelle version) lorsque la ressource est réenregistrée.

<!ELEMENT checked-in (href)>

3.2.2 DAV:auto-version

Si la valeur DAV:auto-version est DAV:checkout-checkin, lorsque une demande de modification (telle que PUT/PROPPATCH) est appliquée à une ressource de version contrôlée enregistrée, la demande est automatiquement précédée d'un désenregistrement et suivie par une opération d'enregistrement.

Si la valeur DAV:auto-version est DAV:checkout-unlocked-checkin, lorsque une demande de modification est appliquée à une ressource de version contrôlée enregistrée, la demande est automatiquement précédée par une opération de dés-enregistrement. Si la ressource n'est pas verrouillée en écriture, la demande est automatiquement suivie par une opération d'enregistrement.

Si la valeur DAV:auto-version est DAV:checkout, lorsque une demande de modification est appliquée à une ressource de version contrôlée enregistrée, la demande est automatiquement précédée par une opération de désenregistrement.

Si la valeur DAV:auto-version est DAV:locked-checkout, lorsque une demande de modification est appliquée à une ressource de version contrôlée enregistrée verrouillée en écriture, la demande est automatiquement précédée par une opération de désenregistrement.

Si une mise à jour d'une ressource enregistrée verrouillée en écriture est automatiquement précédée d'un désenregistrement de cette ressource, le désenregistrement est associé au verrouillage en écriture. Lorsque ce verrouillage en écriture est retiré (par exemple à la suite d'un UNLOCK ou d'une fin de temporisation de verrouillage) si la ressource n'a pas encore été enregistrée, la suppression d'un verrouillage en écriture est automatiquement précédée d'une opération d'enregistrement.

Un serveur PEUT refuser de permettre la modification de la valeur de la propriété DAV:auto-version, ou PEUT ne prendre en charge que les valeurs d'un sous ensemble des valeurs valides.

<!ELEMENT auto-version (checkout-checkin | checkout-unlocked-checkin | checkout | locked-checkout)? >
<!ELEMENT checkout-checkin VIDE>
<!ELEMENT checkout-unlocked-checkin VIDE>
<!ELEMENT checkout VIDE>

<!ELEMENT locked-checkout VIDE>

3.3 Propriétés de ressource vérifiée

La caractéristique version-control introduit les propriétés EXIGÉES suivantes pour une ressource désenregistrée.

3.3.1 DAV:checked-out (protégé)

Cette propriété identifie la version qui a été identifiée par la propriété DAV:checked-in au moment où la ressource a été désenregistrée. Cette propriété est retirée lorsque la ressource est enregistrée.

<!ELEMENT désenregistré (href)>

3.3.2 DAV:predecessor-set

Cette propriété détermine la propriété DAV:predecessor-set *(ensemble de prédécesseurs)* de la version qui résulte de l'enregistrement de cette ressource.

Un serveur PEUT rejeter les tentatives de modifier le DAV:predecessor-set d'une ressource de version contrôlée.

<!ELEMENT ensemble-de-prédécesseurs (href+)>

3.4 Propriétés de version

La caractéristique version-control introduit les propriétés EXIGÉES suivantes pour une version.

3.4.1 DAV:predecessor-set (protégé)

Cette propriété identifie chaque prédécesseur de cette version. Sauf pour la version racine, qui n'a pas de prédécesseur, chaque version a au moins un prédécesseur.

<!ELEMENT ensemble-de-prédécesseurs (href*)>

3.4.2 DAV:successor-set (calculé)

Cette propriété identifie chaque version dont le DAV:predecessor-set identifie cette version.

<!ELEMENT ensemble-de-successeurs (href*)>

3.4.3 DAV:checkout-set (calculé)

Cette propriété identifie chaque ressource désenregistrée dont la propriété DAV:checked-out identifie cette version.

<!ELEMENT ensemble-désenregistré (href*)>

3.4.4 DAV:version-name (protégé)

Cette propriété contient une chaîne définie par le serveur qui est différente pour chaque version dans un certain historique de version. Cette chaîne est destinée à l'affichage à un utilisateur, à la différence de l'URL d'une version, qui n'est normalement utilisé que par un client et n'est pas affiché à l'usager.

<!ELEMENT nom-de-version (#PCDATA)>
valeur de PCDATA : chaîne

3.5 Méthode de VERSION-CONTROL

Une demande VERSION-CONTROL peut être utilisée pour créer une ressource de version contrôlée à l'URL de demande.

Elle peut être appliquée à une ressource versionisable ou à une ressource de version contrôlée.

Si l'URL de demande identifie une ressource versionisable, une nouvelle ressource d'historique de version est créée, une nouvelle version est créée dont le contenu et les propriétés mortes sont copiés de la ressource versionisable, et la ressource reçoit une propriété DAV:checked-in qui est initialisée pour identifier cette nouvelle version.

Si l'URL de demande identifie une ressource de version contrôlée, la ressource reste juste sous le contrôle de version. Cela permet à un client de ne pas savoir si un serveur a automatiquement mis ou non une ressource sous contrôle de version lorsque elle est créée.

Si une demande VERSION-CONTROL échoue, l'état de serveur qui précède la demande DOIT être restauré.

Surveillance : Si un corps de demande est inclus, il DOIT être un élément XML DAV:version-control.

<!ELEMENT contrôle de version TOUT>

Si un corps de réponse pour une demande réussie est inclus, il DOIT être un élément XML DAV:version-control-response. Noter que le présent document ne définit aucun élément pour le corps de réponse VERSION-CONTROL, mais l'élément DAV:version-control-response est défini pour assurer l'interopérabilité entre les futures extensions qui définiront les éléments du corps de réponse VERSION-CONTROL.

<!ELEMENT réponse-de-contrôle-de-version TOUT>

Postconditions :
(DAV:put-under-version-control) : si l'URL de demande identifiait une ressource versionisable au moment de la demande, la demande DOIT avoir créé un nouvel historique de version et DOIT avoir créé une nouvelle ressource de version dans cet historique de version. La ressource DOIT avoir une propriété DAV:checked-in qui identifie la nouvelle version. Le contenu, les propriétés mortes, et le DAV:resourcetype de la nouvelle version DOIVENT être les mêmes que ceux de la ressource. Noter qu'une mise en œuvre peut choisir de localiser l'historique de version et les ressources de version partout où elle le souhaite. En particulier, elle pourrait les localiser sur le même hôte et serveur que la ressource de version contrôlée, sur un hôte virtuel différent entretenu par le même serveur, sur le même hôte, entretenu par un serveur différent, ou sur un hôte différent entretenu par un serveur différent.

(DAV:must-not-change-existing-checked-in-out) : Si l'URL de demande identifiait une ressource déjà sous le contrôle de version au moment de la demande, la demande NE DOIT PAS changer la propriété DAV:checked-in ou DAV:checked-out de cette ressource de version contrôlée.

3.5.1 Exemple de VERSION-CONTROL

```
>>REQUEST
VERSION-CONTROL /foo.html HTTP/1.1
Host: www.webdav.org
Content-Length: 0

>>RESPONSE
HTTP/1.1 200 OK
```

Dans cet exemple, /foo.html est mis sous contrôle de version. Un nouvel historique de version est créé pour lui, et une nouvelle version est créée qui a une copie du contenu et des propriétés mortes de /foo.html. La propriété DAV:checked-in de /foo.html identifie cette nouvelle version.

3.6 Méthode REPORT

Une demande REPORT est un mécanisme extensible pour obtenir des informations sur une ressource. À la différence d'une propriété de ressource, qui a une seule valeur, la valeur d'un rapport peut dépendre d'informations supplémentaires spécifiées dans le corps de demande REPORT et dans les en-têtes de demande REPORT.

Surveillance :
Le corps d'une demande REPORT spécifie quel rapport est demandé, ainsi que toutes informations supplémentaires qui vont être utilisées pour personnaliser le rapport.

La demande PEUT inclure un en-tête Depth. Si aucun en-tête Depth n'est inclus, Depth:0 est supposé.

Le corps de la réponse pour une demande réussie DOIT contenir le rapport demandé.

Si un en-tête de demande Depth est inclus, la réponse DOIT être un 207 Multi-états. La demande DOIT être appliquée séparément à la collection elle-même et à tous les membres de la collection qui satisfont la valeur de Depth. L'élément DAV:prop d'une réponse DAV pour une certaine ressource DOIT contenir le rapport demandé pour cette ressource.

Préconditions :
(DAV:supported-report) : le rapport spécifié DOIT être pris en charge par la ressource identifiée par l'URL de demande.

Postconditions : (DAV:no-modification) : La méthode REPORT NE DOIT PAS avoir changé le contenu ou les propriétés mortes d'une ressource.

3.7 Rapport DAV:version-tree

Le rapport DAV:version-tree décrit les propriétés demandées de toutes les versions dans l'historique de version d'une version. Si le rapport est demandé pour une ressource de version contrôlée, il est redirigé sur sa version DAV:checked-in ou DAV:checked-out.
Le rapport DAV:version-tree DOIT être pris en charge par toutes les ressources de version et toutes les ressources à version contrôlée.

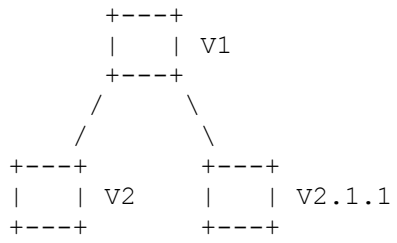
Surveillance : Le corps de demande DOIT être un élément XML DAV:version-tree.
<!ELEMENT version-tree TOUT>
valeur de ANY : séquence de zéro, un ou plusieurs éléments, avec au plus une élément DAV:prop.
prop : voir la RFC 2518, paragraphe 12.11

Le corps de réponse pour une demande réussie DOIT être un élément XML DAV:multistatus.
multistatus : voir la RFC 2518, paragraphe 12.9
Le corps de réponse pour une demande réussie de REPORT DAV:version-tree DOIT contenir un élément DAV:response pour chaque version dans l'historique de version de la version identifiée par l'URL de demande.

3.7.1 Exemple de rapport DAV:version-tree

L'historique de version retracé ci-dessous produirait le rapport d'arborescence de version suivant.

Historique de foo.html




```
>>REQUEST
 REPORT /foo.html HTTP/1.1
 Host: www.webdav.org
 Content-Type: text/xml; charset="utf-8"
 Content-Length: xxxx
 <?xml version="1.0" encoding="utf-8" ?>
 <D:version-tree xmlns:D="DAV:">
  <D:prop>
   <D:version-name/>
   <D:creator-displayname/>
   <D:successor-set/>
  </D:prop>
 </D:version-tree>

>>RESPONSE
```

```
HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
<?xml version="1.0" encoding="utf-8" ?>
<D:multistatus xmlns:D="DAV:">
  <D:response>
    <D:href>http://repo.webdav.org/his/23/ver/V1</D:href>
    <D:propstat>
      <D:prop>
        <D:version-name>V1</D:version-name>
        <D:creator-displayname>Fred</D:creator-displayname>
        <D:successor-set>
          <D:href>http://repo.webdav.org/his/23/ver/V2</D:href>
          <D:href>http://repo.webdav.org/his/23/ver/V2.1.1</D:href>
        </D:successor-set>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
  <D:response>
    <D:href>http://repo.webdav.org/his/23/ver/V2</D:href>
    <D:propstat>
      <D:prop>
        <D:version-name>V2</D:version-name>
        <D:creator-displayname>Fred</D:creator-displayname>
        <D:successor-set/>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
  <D:response>
    <D:href>http://repo.webdav.org/his/23/ver/V2.1.1</D:href>
    <D:propstat>
      <D:prop>
        <D:version-name>V2.1.1</D:version-name>
        <D:creator-displayname>Sally</D:creator-displayname>
        <D:successor-set/>
      </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat>
  </D:response>
</D:multistatus>
```

3.8 Rapport DAV:expand-property

De nombreuses valeurs de propriété sont définies comme un DAV:href, ou un ensemble d'éléments DAV:href. Le rapport DAV:expand-property donne un mécanisme pour restituer dans une demande les propriétés provenant des ressources identifiées par ces éléments DAV:href. Ce rapport non seulement diminue le nombre de demandes requis, mais permet aussi au serveur de minimiser le nombre de transactions de lecture séparées requises sur la mémorisation de versionnage sous-jacente.

Le rapport DAV:expand-property DEVRAIT être pris en charge par toutes les ressources qui acceptent la méthode REPORT.

Surveillance : Le corps de demande DOIT être un élément XML DAV:expand-property.

```
<!ELEMENT expand-property (propriété*)>
<!ELEMENT propriété (propriété*)>
<!ATTLIST nom de propriété NMTOKEN #EXIGÉ>
  valeur de nom : une propriété de type d'élément
<!ATTLIST propriété espace de nom NMTOKEN "DAV:">
  valeur d'espace de nom : un espace de nom XML
```

Le corps de réponse pour une demande réussie DOIT être un élément XML DAV:multistatus.
multistatus : voir la RFC 2518, paragraphe 12.9.

Les propriétés rapportées dans les éléments DAV:prop de l'élément DAV:multistatus DOIVENT être celles identifiées par les éléments DAV:property dans l'élément DAV:expand-property. Si il y a des éléments DAV:property incorporés dans un élément DAV:property, alors, chaque DAV:href dans la valeur de la propriété correspondante est remplacée par un élément DAV:response dont les éléments DAV:prop rapportent les valeurs des propriétés identifiées par les éléments DAV:property incorporés. Les éléments incorporés DAV:property peuvent à leur tour contenir des éléments DAV:property, de sorte que plusieurs niveaux d'expansion de DAV:href peuvent être demandés.

Noter qu'un analyseur de validation DOIT savoir que le rapport DAV:expand-property modifie effectivement le DTD de toute propriété en remplaçant chaque occurrence de "href" dans le DTD par "href | response".

3.8.1 Exemple de DAV:expand-property

Cet exemple décrit comment interroger une ressource de version contrôlée pour déterminer le DAV:creator-display-name et le DAV:activity-set de chaque version dans l'historique de version de cette ressource de version contrôlée. Cet exemple suppose que le serveur prenne en charge la caractéristique d'historique de version (voir la Section 5).

```
>>REQUEST
REPORT /foo.html HTTP/1.1
Host: www.webdav.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
<?xml version="1.0" encoding="utf-8" ?>
<D:expand-property xmlns:D="DAV:">
<D:property name="version-history">
       <D:property name="version-set">
         <D:property name="creator-displayname"/>
         <D:property name="activity-set"/>
       </D:property>
     </D:property>
   </D:expand-property>

>>RESPONSE
HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:multistatus xmlns:D="DAV:">
     <D:response>
       <D:href>http://www.webdav.org/foo.html</D:href>
       <D:propstat>
         <D:prop>
           <D:version-history>
             <D:response>
               <D:href>http://repo.webdav.org/his/23</D:href>
               <D:propstat>
                 <D:prop>
                   <D:version-set>
                     <D:response>
 <D:href>http://repo.webdav.org/his/23/ver/1</D:href>
                       <D:propstat>
                         <D:prop>
 <D:creator-displayname>Fred</D:creator-displayname>
                           <D:activity-set> <D:href>
                             http://www.webdav.org/ws/dev/sally
                           </D:href> </D:activity-set> </D:prop>
                         <D:status>HTTP/1.1 200 OK</D:status>
                       </D:propstat> </D:response>
                     <D:response>
```

```
<D:href>http://repo.webdav.org/his/23/ver/2</D:href>
                <D:propstat>
                  <D:prop>
<D:creator-displayname>Sally</D:creator-displayname>
                    <D:activity-set>
<D:href>http://repo.webdav.org/act/add-refresh-cmd</D:href>
                    </D:activity-set> </D:prop>
                  <D:status>HTTP/1.1 200 OK</D:status>
                </D:propstat> </D:response>
              </D:version-set> </D:prop>
            <D:status>HTTP/1.1 200 OK</D:status>
          </D:propstat> </D:response>
        </D:version-history> </D:prop>
      <D:status>HTTP/1.1 200 OK</D:status>
    </D:propstat> </D:response>
  </D:multistatus>
```

Dans cet exemple sont rapportées les propriétés DAV:creator-displayname et DAV:activity-set des versions dans le DAV:version-set du DAV:version-history de http://www.webdav.org/foo.html.

3.9 Sémantique supplémentaire de OPTIONS

Si le serveur supporte la caractéristique version-control, il DOIT inclure "version-control" comme champ dans l'en-tête de réponse DAV à une demande OPTIONS sur toute ressource qui prend en charge toutes propriétés, rapports, ou méthodes de versionnage.

3.10 Sémantique supplémentaire de PUT

Préconditions supplémentaires :
(DAV:cannot-modify-version-controlled-content) *(ne peut pas modifier le contenu de version contrôlée)* : Si l'URL de demande identifie une ressource avec une propriété DAV:checked-in, la demande DOIT échouer sauf si la sémantique de DAV:auto-version désenregistre automatiquement la ressource.

(DAV:cannot-modify-version) : Si l'URL de demande identifie une version, la demande DOIT échouer.

Si la demande créée une nouvelle ressource qui est automatiquement placée sous contrôle de version, toutes les préconditions pour VERSION-CONTROL s'appliquent à la demande.

Postconditions supplémentaires :
(DAV:auto-checkout) : Si la ressource était une ressource de version contrôlée enregistrée dont la propriété DAV:auto-version indique qu'elle devrait être automatiquement désenregistrée mais pas automatiquement enregistrée pour une demande de modification, le serveur DOIT avoir automatiquement désenregistré la ressource avant d'exécuter la demande. En particulier, la valeur de la propriété DAV:checked-out de la ressource DOIT être celle de la propriété DAV:checked-in avant la demande, la propriété DAV:checked-in DOIT avoir été retirée, et la propriété DAV:predecessor-set DOIT être initialisée à la même propriété que DAV:checked-out. Si une partie quelconque de la séquence désenregistrement/mise à jour échoue, l'état provenant de la partie en échec de la demande DOIT être retourné, et l'état du serveur précédant la séquence de demande DOIT être restauré.

(DAV:auto-checkout-checkin) : Si la ressource a une ressource de version contrôlée enregistrée dont la propriété DAV:auto-version indique qu'elle devrait être automatiquement désenregistrée et automatiquement enregistrée pour une demande de modification, le serveur DOIT alors avoir automatiquement désenregistré la ressource avant d'exécuter la demande et l'enregistrer automatiquement après la demande. En particulier, la propriété DAV:checked-in de la ressource DOIT identifier une nouvelle version dont le contenu et les propriétés mortes sont les mêmes que ceux de la ressource. Le DAV:predecessor-set de la nouvelle version DOIT identifier la version identifiée par la propriété DAV:checked-in d'avant la demande. Si une partie quelconque de la séquence désenregistrement/mise à jour/enregistrement a échoué, l'état provenant de la partie échouée de la demande DOIT être retourné, et l'état du serveur qui précédait la séquence de demande DOIT être restauré.

Si la demande créée une nouvelle ressource, celle-ci PEUT avoir été automatiquement placée sous contrôle de version, et toutes les postconditions pour VERSION-CONTROL s'appliquent à la demande.

3.11 Sémantique supplémentaire de PROPFIND

Une demande PROPFIND DAV:allprop NE DEVRAIT retourner aucune des propriétés définies par le présent document. Cela permet à un serveur de versionnage de fonctionner efficacement lorsque un client inexpérimenté, qui ne comprend pas le coût d'une demande de calcul de toutes les possibilités de propriétés vives à un serveur, produit une demande PROPFIND DAV:allprop.

Préconditions supplémentaires :
(DAV:supported-live-propriété) : Si la demande tente d'accéder à une propriété définie par le présent document, la sémantique de cette propriété DOIT être acceptée par le serveur.

3.12 Sémantique supplémentaire de PROPPATCH

Préconditions supplémentaires :
(DAV:cannot-modify-version-controlled-property) : Si la demande tente de modifier une propriété morte, même sémantique que PUT (voir le paragraphe 3.10).

(DAV:cannot-modify-version) : Si la demande tente de modifier une propriété morte, même sémantique que PUT (voir le paragraphe 3.10).

(DAV:cannot-modify-protected-property) : Une tentative de modifier une propriété qui est définie par le présent document comme étant protégée pour cette sorte de ressource, DOIT échouer.

(DAV:supported-live-property) : Une tentative de modifier une propriété définie par le présent document, mais dont la sémantique n'est pas mise en application par le serveur, DOIT échouer. Cela aide à s'assurer qu'un client sera notifié qu'il essaye d'utiliser une propriété dont la sémantique n'est pas acceptée par le serveur.

Postconditions supplémentaires :
(DAV:auto-checkout) : Si la demande a modifié une propriété morte, même sémantique que PUT (voir le paragraphe 3.10).

(DAV:auto-checkout-checkin) : Si la demande a modifié une propriété morte, même sémantique que PUT (voir le paragraphe 3.10).

3.13 Sémantique supplémentaire de DELETE

Préconditions supplémentaires :
(DAV:no-version-delete) : Un serveur PEUT faire échouer une tentative de DELETE *(supprimer)* une version.

Postconditions supplémentaires :
(DAV:update-predecessor-set) : Si une version a été supprimée, le serveur DOIT avoir remplacé toute référence à cette version dans un DAV:predecessor-set par une copie du DAV:predecessor-set de la version supprimée.

3.14 Sémantique supplémentaire de COPY

Préconditions supplémentaires :
Si la demande crée une nouvelle ressource qui est automatiquement placée sous contrôle de version, toutes les préconditions pour VERSION-CONTROL s'appliquent à la demande.

Postconditions supplémentaires :
(DAV:must-not-copy-versioning-property) : Une propriété définie par le présent document NE DOIT PAS avoir été copiée dans la nouvelle ressource créée par cette demande, mais plutôt, cette propriété de la nouvelle ressource DOIT avoir la valeur initiale par défaut qu'elle aurait eue si la nouvelle ressource avait été créée par une méthode autre que de versionnage telle que PUT ou une MKCOL.

(DAV:auto-checkout) : Si la destination est une ressource de version contrôlée, même sémantique que PUT (voir au paragraphe 3.10).

(DAV:auto-checkout-checkin) : Si la destination est une ressource de version contrôlée, même sémantique que PUT (voir au paragraphe 3.10).

(DAV:copy-creates-new-resource) : Si la source d'une COPY est une ressource de version contrôlée ou de version, et si il n'y a pas de ressource à la destination de la COPY, la COPY crée alors une nouvelle ressource de version non contrôlée à la destination de la COPY. La nouvelle ressource PEUT être automatiquement mise sous contrôle de version, mais la ressource de version contrôlée résultante DOIT être associée au nouvel historique de version créé pour cette nouvelle ressource de version contrôlée, et toutes les postconditions pour VERSION-CONTROL s'appliquent à la demande.

3.15 Sémantique supplémentaire de MOVE

Préconditions supplémentaires :
(DAV:cannot-rename-version) : Si l'URL de demande identifie une version, la demande DOIT échouer.

Postconditions supplémentaires :
(DAV:preserve-versioning-property) : Lorsque une ressource est déplacée d'un URL de source à un URL de destination, une propriété définie par ce document DOIT avoir la même valeur à l'URL de destination que celle qu'elle avait à l'URL de source.

3.16 Sémantique supplémentaire de UNLOCK

Noter que cette sémantique s'applique aussi bien à une demande UNLOCK explicite qu'au retrait d'un verrou à cause d'une fin de temporisation. Si une précondition ou une postcondition ne peut pas être satisfaite, la fin de temporisation de verrouillage NE DOIT PAS survenir.

Préconditions supplémentaires :
(DAV:version-history-is-tree) : Si l'URL de demande identifie une ressource de version contrôlée désenregistrée qui va être automatiquement enregistrée lorsque le verrou sera retiré, alors, les versions identifiées par le DAV:predecessor-set de la ressource désenregistrée DOIVENT être des descendants de la version racine de l'historique de version pour la version DAV:checked-out.

Postconditions supplémentaires :
(DAV:auto-checkin) : Si l'URL de demande identifiait une ressource de version contrôlée désenregistrée qui avait été automatiquement désenregistrée à cause de sa propriété DAV:auto-version, la demande DOIT avoir créé une nouvelle version dans l'historique de version de la version DAV:checked-out. La demande DOIT avoir alloué un URL pour la version qui NE DOIT PAS avoir précédemment identifié d'autre ressource, et NE DOIT jamais identifier une ressource autre que cette version. Le contenu, les propriétés mortes, le type de ressource DAV, et le DAV:predecessor-set de la nouvelle version DOIVENT être copiés de la ressource désenregistrée. Le DAV:version-name de la nouvelle version DOIT être réglé à une valeur définie par le serveur distincte de toutes les autres valeurs de DAV:version-name des autres versions dans le même historique de version. La demande DOIT avoir retiré la propriété DAV:checked-out de la ressource de version contrôlée, et DOIT avoir ajouté une propriété DAV:checked-in qui identifie la nouvelle version.

4. Caractéristique CHECKOUT-IN-PLACE

Avec la caractéristique de contrôle de version, le verrouillage WebDAV peut être utilisé pour éviter la prolifération de versions qui résulterait si chaque modification d'une ressource de version contrôlée produisait une nouvelle version. La caractéristique checkout-in-place donne un mécanisme de remplacement qui permet à un client de désenregistrer et enregistrer explicitement une ressource pour créer une nouvelle version.

4.1 Propriétés de version supplémentaires

La caractéristique checkout-in-place introduit les propriétés EXIGÉES suivantes pour une version.

4.1.1 DAV:checkout-fork

Cette propriété contrôle le comportement de CHECKOUT lorsque une version est déjà vérifiée ou a déjà un successeur. Si le DAV:checkout-fork d'une version est DAV:forbidden, une demande CHECKOUT DOIT échouer si il en résulterait que

cette version apparaîtrait dans la propriété DAV:predecessor-set ou DAV:checked-out de plus d'une ressource version ou désenregistrée. Si DAV:checkout-fork est DAV:discouraged, une telle demande CHECKOUT DOIT échouer sauf si DAV:fork-ok est spécifié dans le corps de la demande CHECKOUT.

Un serveur PEUT rejeter les tentatives de modifier le DAV:checkout-fork d'une version.

<!ELEMENT checkout-fork TOUTE>
valeur de ANY : Séquence d'éléments avec au plus un élément DAV:discouraged ou DAV:forbidden.
<!ELEMENT discouraged VIDE>
<!ELEMENT forbidden VIDE>

4.1.2 DAV:checkin-fork

Cette propriété contrôle le comportement de CHECKIN lorsque une version a déjà un successeur. Si le DAV:checkin-fork d'une version est DAV:forbidden, une demande CHECKIN DOIT échouer si il en résulterait que cette version apparaîtrait dans le DAV:predecessor-set de plus d'une version. Si DAV:checkin-fork est DAV:discouraged, une telle demande CHECKIN DOIT échouer sauf si DAV:fork-ok est spécifié dans le corps de demande CHECKIN.

Un serveur PEUT rejeter les tentatives de modifier le DAV:checkout-fork d'une version.

<!ELEMENT checkin-fork TOUTE>
valeur de ANY : Séquence d'éléments avec au plus un élément DAV:discouraged ou DAV:forbidden.
<!ELEMENT discouraged VIDE>
<!ELEMENT forbidden VIDE>

4.2 Propriétés de ressource désenregistrée

La caractéristique checkout-in-place introduit les propriétés EXIGÉES suivantes pour une ressource désenregistrée.

4.2.1 DAV:checkout-fork

Cette propriété détermine la propriétés DAV:checkout-fork de la version qui résulte de l'enregistrement de cette ressource.

4.2.2 DAV:checkin-fork

Cette propriété détermine la propriété DAV:checkin-fork de la version qui résulte de l'enregistrement de cette ressource.

4.3 Méthode CHECKOUT (appliquée à une ressource de version contrôlée)

Une demande CHECKOUT peut être appliquée à une ressource de version contrôlée enregistrée pour permettre des modifications du contenu et des propriétés mortes de cette ressource de version contrôlée.

Si une demande CHECKOUT échoue, l'état de serveur précédant la demande DOIT être restauré.

Surveillance : Si un corps de demande est inclus, il DOIT être un élément XML DAV:checkout.

<!ELEMENT checkout TOUTE>
valeur de TOUTE : Séquence d'éléments avec au plus un élément DAV:fork-ok.
<!ELEMENT fork-ok VIDE>

Si un corps de réponse est inclus pour une demande réussie, il DOIT être un élément XML DAV:checkout-response.

<!ELEMENT checkout-response TOUTE>

La réponse DOIT inclure un en-tête Cache-Control:no-cache.

Préconditions :
(DAV:must-be-checked-in) *(doit être enregistré)* : Si une ressource de version contrôlée est en cours d'enregistrement, elle DOIT avoir une propriété DAV:checked-in.

(DAV:checkout-of-version-with-descendant-is-forbidden) *(le désenregistrement d'une version qui a des descendants est interdit)* : Si la propriété DAV:checkout-fork de la version en cours d'enregistrement est DAV:forbidden, la demande DOIT échouer si une version identifie cette version dans son DAV:predecessor-set.

(DAV:checkout-of-version-with-descendant-is-discouraged) *(le désenregistrement d'une version qui a des descendants est déconseillé)* : Si la propriété DAV:checkout-fork de la version qui est désenregistrée est DAV:discouraged, la demande DOIT échouer si une version identifie cette version dans son DAV:predecessor-set sauf si DAV:fork-ok est spécifié dans le corps de la demande.

(DAV:checkout-of-checked-out-version-is-forbidden) *(le désenregistrement d'une version désenregistrée est interdit)* : Si la propriété DAV:checkout-fork de la version désenregistrée est DAV:forbidden, la demande DOIT échouer si une ressource désenregistrée identifie cette version dans sa propriété DAV:checked-out.

(DAV:checkout-of-checked-out-version-is-discouraged) *(le désenregistrement d'une version désenregistrée est déconseillé)* : Si la propriété DAV:checkout-fork de la version désenregistrée est DAV:discouraged, la demande DOIT échouer si une ressource désenregistrée identifie cette version dans sa propriété DAV:checked-out sauf si DAV:fork-ok est spécifié dans le corps de la demande.

Postconditions :
(DAV:is-checked-out) : La ressource désenregistrée DOIT avoir une propriété DAV:checked-out qui identifie la version DAV:checked-in précédant le désenregistrement. La ressource de version contrôlée NE DOIT PAS avoir de propriété DAV:checked-in.

(DAV:initialize-predecessor-set) *(initialise l'ensemble de prédécesseurs)* : La propriété DAV:predecessor-set de la ressource désenregistrée DOIT être initialisée à la version DAV:checked-out.

4.3.1 Exemple - CHECKOUT d'une ressource de version contrôlée

```
>>REQUEST
  CHECKOUT /foo.html HTTP/1.1
  Host: www.webdav.org
  Content-Length: 0

>>RESPONSE
  HTTP/1.1 200 OK
  Cache-Control: no-cache
```

Dans cet exemple, la ressource de version contrôlée /foo.html est désenregistrée.

4.4 Méthode CHECKIN (appliquée à une ressource de version contrôlée)

Une demande CHECKIN peut être appliquée à une ressource de version contrôlée désenregistrée pour produire une nouvelle version dont le contenu et les propriétés mortes sont copiés de la ressource désenregistrée.

Si une demande CHECKIN échoue, l"état du serveur précédant la demande DOIT être restauré.

Surveillance : Si un corps de demande est inclus , il DOIT être un élément XML DAV:checkin.

```
<!ELEMENT checkin TOUTE>
  valeur de TOUTE : Séquence d'éléments avec au plus un élément DAV:keep-checked-out et au plus un élément
                    DAV:fork-ok.
<!ELEMENT keep-checked-out VIDE>
<!ELEMENT fork-ok VIDE>
```

Si un corps de réponse est inclus pour une demande réussie, il DOIT être un élément XML DAV:checkin-response.

```
<!ELEMENT checkin-response TOUTE>
```

La réponse DOIT inclure un en-tête Cache-Control:no-cache.

Préconditions :
(DAV:must-be-checked-out) : L'URL de demande DOIT identifier une ressource avec une propriété DAV:checked-out.

(DAV:version-history-is-tree) : Les versions identifiées par le DAV:predecessor-set de la ressource désenregistrée DOIVENT être des descendants de la version racine de l'historique de version de la version DAV:checked-out.

(DAV:checkin-fork-forbidden) : Une demande CHECKIN DOIT échouer si elle causerait une version dont le DAV:checkin-fork serait DAV:forbidden à apparaître dans le DAV:predecessor-set de plus d'une version.

(DAV:checkin-fork-discouraged) : Une demande CHECKIN DOIT échouer si elle causerait une version dont le DAV:checkin-fork serait DAV:discouraged à apparaître dans le DAV:predecessor-set de plus d'une version, sauf si DAV:fork-ok est spécifié dans le corps de demande.

Postconditions :
(DAV:create-version) : La demande DOIT avoir créé une nouvelle version dans l'historique de version de la version DAV:checked-out. La demande DOIT avoir alloué un nouvel URL distinct pour la nouvelle version, et cet URL NE DOIT jamais identifier une ressource autre que cette version. L'URL pour la nouvelle version DOIT être retourné dans un en-tête Réponse de localisation.

(DAV:initialize-version-content-and-properties) : Le contenu, les propriétés mortes, le type de ressource DAV, et le DAV:predecessor-set de la nouvelle version DOIVENT être copiés de la ressource désenregistrée. Le nom de version DAV de la nouvelle version DOIT être réglé à une valeur définie par le serveur distincte de toutes les autres valeurs de DAV:version-name des autres versions dans le même historique de version.

(DAV:checked-in) : Si l'URL de demande identifie une ressource de version contrôlée et si DAV:keep-checked-out n'est pas spécifié dans le corps de demande, la propriété DAV:checked-out de la ressource de version contrôlée DOIT avoir été retirée et une propriété DAV:checked-in qui identifie la nouvelle version DOIT avoir été ajoutée.

(DAV:keep-checked-out) : Si DAV:keep-checked-out est spécifié dans le corps de demande, la propriété DAV:checked-out de la ressource désenregistrée DOIT avoir été mise à jour pour identifier la nouvelle version.

4.4.1 Exemple - CHECKIN

```
>>REQUEST
   CHECKIN /foo.html HTTP/1.1
   Host: www.webdav.org
   Content-Length: 0

>>RESPONSE
   HTTP/1.1 201 Created
   Location: http://repo.webdav.org/his/23/ver/32
   Cache-Control: no-cache
```

Dans cet exemple, la ressource de version contrôlée /foo.html est enregistrée, et une nouvelle version est créée à http://repo.webdav.org/his/23/ver/32.

4.5 Méthode UNCHECKOUT

Une demande UNCHECKOUT peut être appliquée à une ressource de version contrôlée désenregistrée pour annuler l'état CHECKOUT et restaurer l'état pré-CHECKOUT de la ressource de version contrôlée.

Si une demande UNCHECKOUT échoue, le serveur DOIT défaire tous les effets partiels de la demande UNCHECKOUT.

Surveillance : Si un corps de demande est inclus, il DOIT être un élément XML DAV:uncheckout.

```
<!ELEMENT uncheckout TOUTE>
```

Si un corps de réponse est inclus pour une demande réussie, il DOIT être un élément XML DAV:uncheckout-response.

<!ELEMENT uncheckout-response TOUTE>

La réponse DOIT inclure un en-tête Cache-Control:no-cache.

Préconditions : (DAV:must-be-checked-out-version-controlled-resource) : L'URL de demande DOIT identifier une ressource de version contrôlée avec une propriété DAV:checked-out.

Postconditions :
(DAV:cancel-checked-out) : La valeur de la propriété DAV:checked-in est celle de la propriété DAV:checked-out avant la demande, et la propriété DAV:checked-out a été retirée.

(DAV:restore-content-and-dead-properties) : Le contenu et les propriétés mortes de la ressource de version contrôlée sont copiés de sa version DAV:checked-in.

4.5.1 Exemple - UNCHECKOUT

```
>>REQUEST
   UNCHECKOUT /foo.html HTTP/1.1
   Host: www.webdav.org
   Content-Length: 0

>>RESPONSE
   HTTP/1.1 200 OK
   Cache-Control: no-cache
```

Dans cet exemple, le contenu et les propriétés mortes de la ressource de version contrôlée identifiée par http://www.webdav.org/foo.html sont restaurés à leur valeurs précédant le plus récent CHECKOUT de cette ressource de version contrôlée.

4.6 Sémantique supplémentaire de OPTIONS

Si un serveur prend en charge la caractéristique checkout-in-place, il DOIT inclure "checkout-in-place" comme champ dans l'en-tête de réponse DAV d'une demande OPTIONS sur toute ressource qui accepte des propriétés, des rapports, ou des méthodes de gestion de version.

5. Caractéristique Version-History

Il est souvent utile d'avoir accès à un historique de version même après la suppression de toutes les ressources de version contrôlées pour cet historique de version. Un serveur peut fournir cette fonctionnalité en prenant en charge les ressources d'historique de version. Une ressource d'historique de version est une ressource qui existe dans un espace de noms défini par un serveur et est donc non affectée par une suppression ou un déplacement de ressources de version contrôlée. Une ressource d'historique de version est un endroit approprié pour ajouter une propriété qui s'applique logiquement à tous les états d'une ressource. Le rapport DAV:expand-property (voir au paragraphe 3.8) peut être appliqué au DAV:version-set d'une ressource d'historique de version pour fournir divers rapports utiles sur toutes les versions de cet historique de version.

5.1 Propriétés d'historique de version

Le type de ressource DAV d'un historique de version DOIT être DAV:version-history.

La caractéristique version-history introduit les propriétés EXIGÉES suivantes pour un historique de version.

5.1.1 DAV:version-set (protégé)

Cette propriété identifie chaque version de cet historique de version.

<!ELEMENT version-set (href+)>

5.1.2 DAV:root-version (calculé)

Cette propriété identifie la version racine de cet historique de version.

```
<!ELEMENT root-version (href)>
```

5.2 Propriétés supplémentaires de ressource de version contrôlée

La caractéristique d'historique de version introduit la propriété EXIGÉE suivante pour une ressource de version contrôlée.

5.2.1 DAV:version-history (calculé)

Cette propriété identifie la ressource d'historique de version pour la version DAV:checked-in ou DAV:checked-out de cette ressource de version contrôlée.

```
<!ELEMENT version-history (href)>
```

5.3 Propriétés de version supplémentaires

La caractéristique version-history introduit la propriété EXIGÉE suivante pour une version.

5.3.1 DAV:version-history (calculé)

Cette propriété identifie l'historique de version qui contient cette version.

```
<!ELEMENT version-history (href)>
```

5.4 Rapport DAV:locate-by-history

De nombreuses propriétés identifient une version à partir d'un historique de version. Il est souvent utile d'être capable de localiser efficacement une ressource de version contrôlée pour cet historique de version. Le rapport DAV:locate-by-history peut être appliqué à une collection pour localiser le membre de la collection qui est une ressource de version contrôlée pour une ressource d'historique de version spécifiée.

Surveillance : Le corps de demande DOIT être un élément XML DAV:locate-by-history.

```
<!ELEMENT locate-by-history (version-history-set, prop)>
<!ELEMENT version-history-set (href+)>
  prop : voir la RFC 2518, paragraphe 12.11
```

Le corps de réponse pour une demande réussie DOIT être un élément XML DAV:multistatus contenant chaque ressource de version contrôlée qui est membre de la collection identifiée par l'URL de demande, et dont la propriété DAV:version-history identifie une des ressources d'historique de version identifiée par le corps de demande. L'élément DAV:prop dans le corps de demande identifie quelles propriétés devraient être rapportées dans les éléments DAV:prop dans le corps de réponse.

Préconditions : (DAV:must-be-version-history) : Chaque membre de l'élément DAV:version-history-set dans le corps de demande DOIT identifier une ressource d'historique de version.

5.4.1 Exemple – rapport DAV:locate-by-history

```
>>REQUEST
   REPORT /ws/public HTTP/1.1
   Host: www.webdav.org
   Content-Type: text/xml; charset="utf-8"
   Content-Length: xxxx

    <?xml version="1.0" encoding="utf-8" ?>
    <D:locate-by-history xmlns:D="DAV:">
      <D:version-history-set>
```

```
      <D:href>http://repo.webdav.org/his/23</D:href>
      <D:href>http://repo.webdav.org/his/84</D:href>
      <D:href>http://repo.webdav.org/his/129</D:href>
    <D:version-history-set/>
    <D:prop>
      </D:version-history>
    </D:prop>
  </D:locate-by-history>

>>RESPONSE
  HTTP/1.1 207 Multi-Status
  Content-Type: text/xml; charset="utf-8"
  Content-Length: xxxx

  <?xml version="1.0" encoding="utf-8" ?>
  <D:multistatus xmlns:D="DAV:">
    <D:response>
      <D:href>http://www.webdav.org/ws/public/x/test.html</D:href>
      <D:propstat>
        <D:prop>
          <D:version-history>
            <D:href>http://repo.webdav.org/his/23</D:href>
          </D:version-history>
        </D:prop>
        <D:status>HTTP/1.1 200 OK</D:status>
      </D:propstat>
    </D:response>
  </D:multistatus>
```

Dans cet exemple, il y a seulement un membre de version contrôlée de /ws/public qui est une ressource de version contrôlée pour une des trois ressources d'historique de version spécifiées. En particulier, /ws/public/x/test.html est la ressource de version contrôlée pour http://repo.webdav.org/his/23.

5.5 Sémantique supplémentaire de OPTIONS

Si le serveur accepte la caractéristique version-history, il DOIT inclure "version-history" comme champ dans l'en-tête de réponse DAV d'une demande OPTIONS sur toute ressource qui prend en charge des propriétés, des rapports, ou des méthodes de gestion de version.

Un élément DAV:version-history-collection-set PEUT être inclus dans le corps de demande pour identifier les collections qui peuvent contenir des ressources d'historique de version.

Surveillance supplémentaire :
Si un corps de demande XML est inclus, il DOIT être un élément XML DAV:options.

```
<!ELEMENT options TOUTE>
  valeur de TOUTE : Séquence d'éléments avec au plus un élément DAV:version-history-collection-set.
```

Si un corps de réponse XML pour une demande réussie est inclus, il DOIT être un élément XML DAV:options-response.

```
<!ELEMENT options-response TOUTE>
  valeur de TOUTE : Séquence d'éléments avec au plus un élément DAV:version-history-collection-set.
<!ELEMENT version-history-collection-set (href*)>
```

Si DAV:version-history-collection-set est inclus dans le corps de demande, le corps de réponse pour une demande réussie DOIT contenir un élément DAV:version-history-collection-set qui identifie les collections qui peuvent contenir les historiques de version. Une collection identifiée PEUT être la collection racine d'une arborescence de collections, dont toutes peuvent contenir des historiques de version. Comme différents serveurs peuvent contrôler des parties différentes de l'espace de nom d'URL, différentes ressources sur le même hôte PEUVENT avoir des valeurs différentes de DAV:version-history-collection-set. Les collections identifiées PEUVENT être localisées sur des hôtes différents à partir de la ressource.

5.6 Sémantique supplémentaire de DELETE

Postconditions supplémentaires :
(DAV:delete-version-set) : Si la demande supprimait un historique de version, la demande DOIT avoir supprimé toutes les versions dans le DAV:version-set de cet historique de version, et DOIT avoir satisfait les postconditions pour la suppression de version (voir au paragraphe 3.13).

(DAV:version-history-has-root) : Si la demande supprimait la version racine d'un historique de version, la demande DOIT avoir mis à jour la DAV:root-version de l'historique de version pour se référer à une autre version qui est un ancêtre de toutes les autres versions restantes dans cet historique de version. Un résultat de cette postcondition est que chaque historique de version va avoir au moins une version, et que la seule façon de supprimer toutes les versions est de supprimer la ressource d'historique de version.

5.7 Sémantique supplémentaire de COPY

Préconditions supplémentaires :
(DAV:cannot-copy-history) : Si l'URL de demande identifie un historique de version, la demande DOIT échouer. Afin de créer un autre historique de version dont les versions ont le même contenu et les mêmes propriétés mortes, la séquence appropriée de demandes VERSION-CONTROL, CHECKOUT, PUT, PROPPATCH, et CHECKIN doit être faite.

5.8 Sémantique supplémentaire de MOVE

Préconditions supplémentaires :
(DAV:cannot-rename-history) : Si l'URL de demande identifie un historique de version, la demande DOIT échouer.

5.9 Sémantique supplémentaire de VERSION-CONTROL

Postconditions supplémentaires :
(DAV:new-version-history) : Si la demande créait un nouvel historique de version, la demande DOIT avoir alloué un nouvel URL défini par le serveur pour cet historique de version qui NE DOIT PAS avoir précédemment identifié d'autre ressource, et NE DOIT jamais identifier une ressource autre que cet historique de version.

5.10 Sémantique supplémentaire de CHECKIN

Postconditions supplémentaires : (DAV:add-to-history) : Un URL pour la nouvelle ressource de version DOIT avoir été ajouté au DAV:version-set de l'historique de version de la version DAV:checked-out.

6. Caractéristique Workspace

Pour permettre que plusieurs utilisateurs travaillent concurremment à ajouter des versions au même historique de version, il est nécessaire d'allouer sur le serveur plusieurs ressources désenregistrées pour le même historique de version. Même si un seul utilisateur fait des changements à une ressource, cet utilisateur va parfois souhaiter créer une version "privée", et exposer cette version ultérieurement. Une façon de fournir cette fonctionnalité dépend de ce que le client garde trace de son ensemble actuel de ressources désenregistrées. C'est la caractéristique working-ressource définie à la Section 8. L'autre façon de fournir cette fonctionnalité évite d'avoir besoin d'un état persistant chez le client, et d'avoir plutôt le serveur qui conserve un espace de nom significatif pour l'homme pour les ensembles de ressources désenregistrées qui s'y rapportent. C'est la caractéristique workspace *(espace de travail)* définie dans cette section.

La caractéristique workspace introduit une "ressource de travail". Une ressource de travail est une collection dont les membres sont des ressources en rapport de version contrôlée et de version non contrôlées. Plusieurs espaces de travail peuvent être utilisés pour exposer concurremment différentes versions et configurations d'un ensemble de ressources de version contrôlée. Afin de rendre des changements à une ressource de version contrôlée dans un espace de travail visibles dans un autre espace de travail, cette ressource de version contrôlée doit être enregistrée, et ensuite la ressource de version contrôlée correspondante dans l'autre espace de travail peut être mise à jour pour afficher le contenu et les propriétés mortes de la nouvelle version.

Pour assurer une fusion non ambiguë (voir la Section 11) et la sémantique de base (voir la Section 12) un espace de travail peut contenir au plus une ressource de version contrôlée pour un historique de version donné. Ceci est nécessaire pour une fusion non ambiguë parce que la méthode MERGE doit identifier quelle ressource de version contrôlée est la cible de la fusion pour une certaine version. Ceci est exigé pour une ligne de base non ambiguë parce que une ligne de base ne peut choisir qu'une version pour une certaine ressource de version contrôlée.

Initialement, on peut créer un espace de travail vide. Des ressources de version non contrôlées peuvent alors être ajoutées à l'espace de travail avec des demandes WebDAV standard telles que PUT et MKCOL. Des ressources de version contrôlée peuvent être ajoutées à l'espace de travail avec des demandes VERSION-CONTROL. Si la caractéristique de ligne de base est prise en charge, les collections dans l'espace de travail peuvent être placées sous le contrôle de la ligne de base, et ensuite initialisées par les lignes de base existantes.

6.1 Propriétés de Workspace

La caractéristique workspace introduit les propriété EXIGÉES suivantes pour un espace de travail.

6.1.1 DAV:workspace-checkout-set (calculé)

Cette propriété identifie chaque ressource désenregistrée dont la propriété DAV:workspace identifie cet espace de travail.

```
<!ELEMENT workspace-checkout-set (href*)>
```

6.2 Propriétés de ressource supplémentaires

La caractéristique workspace introduit la propriété EXIGÉE suivante pour une ressource WebDAV.

6.2.1 DAV:workspace (protégé)

La propriété DAV:workspace d'une ressource workspace DOIT s'identifier elle-même. La propriété DAV:workspace de tout autre type de ressource DOIT être la même que la DAV:workspace de sa collection parente.

```
<!ELEMENT workspace (href)>
```

6.3 Méthode MKWORKSPACE

Une demande MKWORKSPACE crée une nouvelle ressource d'espace de travail. Un serveur PEUT restreindre la création d'espace de travail à des collections particulières, mais un client peut déterminer la localisation de ces collections à partir d’une demande OPTIONS DAV:workspace-collection-set (voir au paragraphe 6.4).

Si une demande MKWORKSPACE échoue, l'état du serveur précédant la demande DOIT être restauré.

Surveillance :
Si un corps de demande est inclus, il DOIT être un élément XML DAV:mkworkspace.

```
<!ELEMENT mkworkspace TOUTE>
```

Si un corps de réponse pour une demande réussie est inclus, il DOIT être un élément XML DAV:mkworkspace-response.

```
<!ELEMENT mkworkspace-response TOUTE>
```

La réponse DOIT inclure un en-tête Cache-Control:no-cache.

Préconditions :
(DAV:ressource-must-be-null) : une ressource NE DOIT PAS exister à l'URL de demande.

(DAV:workspace-location-ok) : l'URL de demande DOIT identifier une localisation où un espace de travail peut être créé.

Postconditions :

(DAV:initialize-workspace) : un nouvel espace de travail existe à l'URL de demande. Le type de ressource DAV: de l'espace de travail DOIT être DAV:collection. Le DAV:workspace de l'espace de travail DOIT identifier l'espace de travail.

6.3.1 Exemple - MKWORKSPACE

```
>>REQUEST
MKWORKSPACE /ws/public HTTP/1.1
Host: www.webdav.org
Content-Length: 0

>>RESPONSE
HTTP/1.1 201 Created
Cache-Control: no-cache
```

Dans cet exemple, un nouvel espace de travail est créé à http://www.webdav.org/ws/public.

6.4 Sémantique supplémentaire de OPTIONS

Si un serveur prend en charge la caractéristique workspace, il DOIT inclure un champ "workspace" dans l'en-tête de réponse DAV d'une demande OPTIONS sur toute ressource qui prend en charge des propriétés, rapports, ou méthodes de gestion de version.

Si un serveur prend en charge la caractéristique workspace, il DOIT aussi prendre en charge la caractéristique checkout-in-place et la caractéristique version-history.

Un élément DAV:workspace-collection-set PEUT être inclus dans le corps de demande pour identifier des collections qui peuvent contenir des ressources d'espace de travail.

Surveillance supplémentaire :
Si un corps de demande XML est inclus, il DOIT être un élément XML DAV:options.

```
<!ELEMENT options TOUTE>
  valeur de TOUTE : séquence d'éléments avec au plus un élément DAV:workspace-collection-set.
```

Si un corps de réponse XML pour une demande réussie est inclus, il DOIT être un élément XML DAV:options-response.

```
<!ELEMENT options-response TOUTE>
  valeur de TOUTE : séquence d'éléments avec au plus un élément DAV:workspace-collection-set.
<!ELEMENT workspace-collection-set (href*)>
```

Si DAV:workspace-collection-set est inclus dans le corps de demande, le corps de réponse pour une demande réussie DOIT contenir un élément DAV:workspace-collection-set qui identifie les collections qui peuvent contenir des espaces de travail. Une collection identifiée PEUT être la collection racine d'une arborescence de collections, qui peuvent toutes contenir des espaces de travail. Comme différents serveurs peuvent contrôler des parties différentes de l'espace de nom d'URL, différentes ressources sur le même hôte PEUVENT avoir des valeurs différentes de DAV:workspace-collection-set. Les collections identifiées PEUVENT être situées sur des hôtes différents pour la ressource.

6.4.1 Exemple - OPTIONS

```
>>REQUEST
OPTIONS /doc HTTP/1.1
Host: www.webdav.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
<?xml version="1.0" encoding="utf-8" ?>
<D:options xmlns:D="DAV:">
<D:version-history-collection-set/>
<D:workspace-collection-set/>
</D:options>

>>RESPONSE
HTTP/1.1 200 OK
DAV: 1
DAV: version-control,checkout-in-place,version-history,workspace
```

```
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
   <?xml version="1.0" encoding="utf-8" ?>
   <D:options-response xmlns:D="DAV:">
    <D:version-history-collection-set>
     <D:href>http://repo.webdav.org/his</D:href>
    </D:version-history-collection-set>
    <D:workspace-collection-set>
     <D:href>http://www.webdav.org/public/ws</D:href>
     <D:href>http://www.webdav.org/private/ws</D:href>
    </D:workspace-collection-set>
   </D:options-response>
```

Dans cet exemple, le serveur indique qu'il fournit la prise en charge de DAV classe 1 et la prise en charge de la gestion de version de base d'espace de travail de serveur. De plus, le serveur indique les localisations demandées des ressources d'historique de version et des ressources d'espace de travail.

6.5 Sémantique supplémentaire de DELETE

Postconditions supplémentaires : (DAV:delete-workspace-members) : Si un espace de travail est supprimé, toute ressource qui identifie cet espace de travail dans sa propriété DAV:workspace DOIT être supprimée.

6.6 Sémantique supplémentaire de MOVE

Postconditions supplémentaires :
(DAV:workspace-member-moved) : Si l'URL de demande n'identifiait pas d'espace de travail, le DAV:workspace de la destination DOIT avoir été mis à jour pour avoir la même valeur que le DAV:workspace de la collection parente de la destination.

(DAV:workspace-moved) : Si l'URL de demande identifiait un espace de travail, toute référence à cet espace de travail dans une propriété DAV:workspace DOIT avoir été mise à jour pour se référer à la nouvelle localisation de cet espace.

6.7 Sémantique supplémentaire de VERSION-CONTROL

Une demande VERSION-CONTROL peut être utilisée pour créer une nouvelle ressource de version contrôlée pour un historique de version existant. Cela permet la création de ressources de version contrôlée pour le même historique de version dans plusieurs espaces de travail.

Surveillance supplémentaire :

```
<!ELEMENT version-control TOUTE>
  valeur de TOUTE : séquence d'éléments avec au  plus un élément DAV:version.
<!ELEMENT version (href)>
```

Préconditions supplémentaires :
(DAV:cannot-add-to-existing-history) : si l'élément de corps de demande DAV:version-control contient un élément DAV:version, l'URL de demande NE DOIT PAS identifier une ressource.

(DAV:must-be-version) : le DAV:href de l'élément DAV:version DOIT identifier une version.

(DAV:one-version-controlled-ressource-per-history-per-workspace) : si le corps de demande DAV:version-control spécifie une version, et si l'URL de demande est un membre d'un espace de travail, alors il NE DOIT PAS être déjà un membre version-controlled de cet espace de travail dont la propriété DAV:checked-in ou DAV:checked-out identifie une version de l'historique de version de la version spécifiée dans le corps de demande.

Postconditions supplémentaires :
(DAV:new-version-controlled-ressource) : si l'URL de demande N'identifie PAS une ressource, une nouvelle ressource de version contrôlée existe à l'URL de demande dont le contenu et les propriétés mortes sont initialisés par ceux de la version dans le corps de demande, et dont la propriété DAV:checked-in identifie cette version.

6.7.1 Exemple - VERSION-CONTROL (utilisant un historique de version existant)

```
>>REQUEST
VERSION-CONTROL /ws/public/bar.html HTTP/1.1
Host: www.webdav.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:version-control xmlns:D="DAV:">
<D:version>
<D:href>http://repo.webdav.org/his/12/ver/V3</D:href>
</D:version>
</D:version-control>

>>RESPONSE
HTTP/1.1 201 Created
Cache-Control: no-cache
```

Dans cet exemple, une nouvelle ressource de version contrôlée est créée à /ws/public/bar.html. Le contenu et les propriétés mortes de la nouvelle ressource de version contrôlée sont initialisés comme étant les mêmes que ceux de la version identifiée par http://repo.webdav.org/his/12/ver/V3.

7. Caractéristique UPDATE

La caractéristique update donne un mécanisme pour changer l'état d'une ressource de version contrôlée enregistrée comme étant celle d'une autre version de l'historique de version de cette ressource.

7.1 Méthode UPDATE

La méthode UPDATE modifie le contenu et les propriétés mortes d'une ressource de version contrôlée enregistrée (la "cible de mise à jour") pour être ceux d'une version spécifiée (la "source de mise à jour") à partir de l'historique de version de cette ressource de version contrôlée.

La réponse à une demande UPDATE identifie les ressources modifiées par la demande, afin qu'un client puisse efficacement mettre à jour tout état qu'il conserve en mémoire tampon. Les extensions à la méthode UPDATE permettent que plusieurs ressources soient modifiées à partir d'une seule demande UPDATE (voir au paragraphe 12.13).

Surveillance : Le corps de demande DOIT être un élément DAV:update.

```
<!ELEMENT update TOUTE>
  valeur de TOUTE : séquence d'éléments avec au plus un élément DAV:version et au plus un élément DAV:prop.
<!ELEMENT version (href)>
  prop : voir la RFC 2518, paragraphe 12.11
```

La réponse pour une demande réussie DOIT être un 207 Multi-Status, où l'élément XML DAV:multistatus dans le corps de réponse identifie toutes les ressources qui ont été modifiées par la demande.

Multistatus : voir la RFC 2518, paragraphe 12.9

La réponse DOIT inclure un en-tête Cache-Control:no-cache.

Postconditions :
(DAV:update-content-and-properties) : Si l'élément DAV:version dans le corps de demande identifiait une version qui est dans le même historique de version que la version DAV:checked-in d'une ressource de version contrôlée identifiée par l'URL de demande, alors le contenu et les propriétés mortes de cette ressource de version contrôlée DOIVENT être les mêmes que ceux de la version spécifiée par l'élément DAV:version, et la propriété DAV:checked-in de la ressource de version contrôlée DOIT identifier cette version. L'URL de demande DOIT apparaître dans un élément DAV:response dans le corps de réponse.

(DAV:report-properties) : Si DAV:prop est spécifié dans le corps de demande, les propriétés spécifiées dans l'élément

DAV:prop DOIVENT être rapportées dans les éléments DAV:response dans le corps de réponse.

7.1.1 Exemple - UPDATE

```
>>REQUEST
UPDATE /foo.html HTTP/1.1
Host: www.webdav.org
Content-type: text/xml; charset="utf-8"
Content-Length: xxxx

<?xml version="1.0" encoding="utf-8" ?>
<D:update xmlns:D="DAV:">
<D:version>
<D:href>http://repo.webdav.org/his/23/ver/33</D:href>
</D:version>
</D:update>

>>RESPONSE
HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
Cache-Control: no-cache

   <?xml version="1.0" encoding="utf-8" ?>
   <D:multistatus xmlns:D="DAV:">
     <D:response>
       <D:href>http://www.webdav.org/foo.html</D:href>
       <D:status>HTTP/1.1 200 OK</D:status>
     </D:response>
```

Dans cet exemple, le contenu et les propriétés mortes de http://repo.webdav.org/his/23/ver/33 sont copiés à la ressource de version contrôlée /foo.html, et la propriété DAV:checked-in de /foo.html est mise à jour pour se référer à http://repo.webdav.org/his/23/ver/33.

7.2 Sémantique supplémentaire de OPTIONS

Si le serveur prend en charge la caractéristique update, il DOIT inclure "update" comme champ dans l'en-tête de réponse DAV à une demande OPTIONS sur toute ressource qui prend en charge des propriétés, rapports, ou méthodes de gestion de version.

8. Caractéristique Label

Une "étiquette" de version est une chaîne qui distingue une version dans un historique de version de toutes les autres versions dans cet historique de version. Une étiquette peut être allouée automatiquement par un serveur, ou elle peut être allouée par un client afin de donner un nom significatif à cette version. Une certaine étiquette de version peut être allouée à au plus une version d'un certain historique de version, mais les étiquettes allouées par le client peuvent être réallouées à tout moment à une autre version. Noter que bien qu'une certaine étiquette puisse être appliquée à au plus une version provenant du même historique de version, la même étiquette peut s'appliquer à des versions provenant d'historiques de version différents.

Pour certaines méthodes, si l'URL de demande identifie une ressource de version contrôlée, une étiquette peut être spécifiée dans un en-tête Demande d'étiquette (voir au paragraphe 8.3) pour causer l'application de la méthode à la version choisie par cette étiquette à partir de l'historique de version de cette ressource de version contrôlée.

8.1 Propriétés supplémentaires de Version

La caractéristique label introduit la propriété EXIGÉE suivante pour une version.

8.1.1 DAV:label-name-set (protégé)

Cette propriété contient les étiquettes qui choisissent actuellement cette version.

```
<!ELEMENT label-name-set (label-name*)>
<!ELEMENT label-name (#PCDATA)>
  valeur de PCDATA : chaîne
```

8.2 Méthode LABEL

Une demande LABEL peut être appliquée à une version pour modifier les étiquettes qui choisissent cette version. La casse du nom d'une étiquette DOIT être préservée lorsque il est mémorisé et restitué. Quand on compare deux noms d'étiquette pour décider si ils correspondent ou non, un serveur DEVRAIT utiliser une comparaison sensible à la casse codée en UTF-8 à échappement d'URL des deux noms d'étiquette.

Si une demande LABEL est appliquée à une ressource de version contrôlée enregistrée, l'opération DOIT être appliquée à la version DAV:checked-in de cette ressource de version contrôlée.

Surveillance : Le corps de demande DOIT être un élément DAV:label.

```
<!ELEMENT label TOUTE>
  valeur de TOUTE : séquence d'éléments avec au plus un élément DAV:add, DAV:set, ou DAV:remove.
<!ELEMENT add (label-name)>
<!ELEMENT set (label-name)>
<!ELEMENT remove (label-name)>
<!ELEMENT label-name (#PCDATA)>
  valeur de PCDATA : chaîne
```

La demande PEUT inclure un en-tête Label.

La demande PEUT inclure un en-tête Depth. Si aucun en-tête Depth n'est inclus, Depth:0 est supposé. La sémantique standard de Depth s'applique, et la demande est appliquée à la collection identifiée par l'URL de demande et à tous les membres de la collection qui satisfont à la valeur Depth. Si un en-tête Depth est inclus et si la demande échoue sur toutes les ressources, la réponse DOIT être 207 Multi-Status qui identifie toutes les ressources pour lesquelles la demande a échoué.

Si un corps de réponse pour une demande réussie est inclus, il DOIT être un élément XML DAV:label-response.

```
<!ELEMENT label-response TOUTE>
```

La réponse DOIT inclure un en-tête Cache-Control:no-cache.

Préconditions : (DAV:must-be-checked-in) : Si l'URL de demande identifie une ressource de version contrôlée, la ressource de version contrôlée DOIT être enregistrée.

(DAV:must-select-version-in-history) : Si un en-tête de demande d'étiquette est inclus et si l'URL de demande identifie une ressource de version contrôlée, l'étiquette spécifiée DOIT choisir une version dans l'historique de version de la ressource de version contrôlée.

(DAV:add-must-be-new-label) : Si DAV:add est spécifié dans le corps de demande, l'étiquette spécifiée NE DOIT PAS apparaître dans le DAV:label-name-set de toute version dans l'historique de version de cette ressource de version contrôlée.

(DAV:label-must-exist) : SI DAV:remove est spécifié dans le corps de demande, l'étiquette spécifiée DOIT apparaître dans le DAV:label-name-set de cette version.

Postconditions : (DAV:add-or-set-label) : Si DAV:add ou DAV:set est spécifié dans le corps de demande, l'étiquette spécifiée DOIT apparaître dans le DAV:label-name-set de la version spécifiée, et NE DOIT PAS apparaître dans le DAV:label-name-set de toute autre version dans l'historique de version de cette version.

(DAV:remove-label) : Si DAV:remove est spécifié dans le corps de demande, l'étiquette spécifiée NE DOIT PAS apparaître dans le DAV:label-name-set de toute version dans l'historique de version de cette version.

8.2.1 Exemple – établissement d'une étiquette

```
>>REQUEST
LABEL /foo.html HTTP/1.1
Host: www.webdav.org
Content-type: text/xml; charset="utf-8"
Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:label xmlns:D="DAV:">
     <D:set>
       <D:label-name>default</D:label-name>
     </D:set>
   </D:label>

>>RESPONSE
HTTP/1.1 200 OK
Cache-Control: no-cache
```

Dans cet exemple, l'étiquette "default" est appliquée à la version DAV:checked-in de /foo.html.

8.3 En-tête d'étiquette

Pour certaines méthodes (par exemple GET, PROPFIND), si l'URL de demande identifie une ressource de version contrôlée, une étiquette peut être spécifiée dans un en-tête Demande d'étiquette pour causer l'application de la méthode à la version choisie par cette étiquette à partir de l'historique de version de cette ressource de version contrôlée.

La valeur d'un en-tête d'étiquette est le nom de l'étiquette, codée en utilisant l'UTF-8 à échappement d'URL. Par exemple, l'étiquette "release B.3" est identifiée par l'en-tête suivant :

```
Label: release%20B.3
```

Un en-tête Label DOIT n'avoir aucun effet sur une demande dont l'URL de demande n'identifie pas une ressource de version contrôlée. En particulier, il DOIT n'avoir aucun effet sur une demande dont l'URL de demande identifie une version ou un historique de version.

Un serveur DOIT retourner un en-tête HTTP-1.1 Vary contenant Label dans une réponse réussie à une demande qu'il est possible de mettre en antémémoire (par exemple, GET) qui inclut un en-tête Label.

8.4 Sémantique supplémentaire de OPTIONS

Si le serveur prend en charge la caractéristique label, il DOIT inclure "label" comme champ dans l'en-tête de réponse DAV à partir d'une demande OPTIONS sur toute ressource qui prend en charge des propriétés, rapports, ou méthodes de gestion de version.

8.5 Sémantique supplémentaire de GET

Surveillance supplémentaire : La demande PEUT inclure un en-tête Label.

Préconditions supplémentaires : (DAV:must-select-version-in-history) : Si un en-tête de demande Label est inclus et si l'URL de demande identifie une ressource de version contrôlée, l'étiquette spécifiée DOIT choisir une version dans l'historique de version de la ressource de version contrôlée.

Postconditions supplémentaires : (DAV:apply-request-to-labeled-version) : Si l'URL de demande identifie une ressource de version contrôlée et si l'en-tête Demande d'étiquette est inclus, la réponse DOIT comprendre le contenu de la version spécifiée plutôt que celui de la ressource de version contrôlée.

8.6 Sémantique supplémentaire de PROPFIND

Surveillance supplémentaire : La demande PEUT inclure un en-tête Label.

Préconditions supplémentaires : (DAV:must-select-version-in-history) : Si un en-tête Demande d'étiquette est inclus et si l'URL de demande identifie une ressource de version contrôlée, l'étiquette spécifiée DOIT choisir une version dans l'historique de version de la ressource de version contrôlée.

Postconditions supplémentaires : (DAV:apply-request-to-labeled-version) : Si l'URL de demande identifie une ressource de version contrôlée et si un en-tête Demande d'étiquette est inclus, la réponse DOIT contenir les propriétés de la version spécifiée plutôt que celles de la ressource de version contrôlée.

8.7 Sémantique supplémentaire de COPY

Surveillance supplémentaire : La demande PEUT inclure un en-tête Label.

Préconditions supplémentaires : (DAV:must-select-version-in-history) : Si un en-tête Demande d'étiquette est inclus et si l'URL de demande identifie une ressource de version contrôlée, l'étiquette spécifiée DOIT choisir une version dans l'historique de version de la ressource de version contrôlée.

Postconditions supplémentaires : (DAV:apply-request-to-labeled-version) : Si l'URL de demande identifie une ressource de version contrôlée et si un en-tête Demande d'étiquette est inclus, la demande DOIT avoir copié les propriétés et le contenu de la version spécifiée plutôt que celles de la ressource de version contrôlée.

8.8 Sémantique supplémentaire de CHECKOUT

Si le serveur prend en charge l'option ressource de travail, un en-tête LABEL peut être inclus pour désenregistrer la version choisie par l'étiquette spécifiée.

Surveillance supplémentaire : La demande PEUT inclure un en-tête Label.

Préconditions supplémentaires :
(DAV:must-select-version-in-history) : Si un en-tête Demande d'étiquette est inclus et si l'URL de demande identifie une ressource de version contrôlée, l'étiquette spécifiée DOIT choisir une version dans l'historique de version de la ressource de version contrôlée.

(DAV:must-not-have-label-and-apply-to-version) : Si un en-tête Demande d'étiquette est inclus, le corps de demande NE DOIT PAS contenir un élément DAV:apply-to-version.

Postconditions supplémentaires : (DAV:apply-request-to-labeled-version) : Si l'URL de demande identifie une ressource de version contrôlée enregistrée, et si un en-tête Demande d'étiquette est inclus, le CHECKOUT DOIT avoir été appliqué à la version choisie par l'étiquette spécifiée, et non à la ressource de version contrôlée elle-même.

8.9 Sémantique supplémentaire de UPDATE

Si le corps de demande d'une demande UPDATE contient un élément DAV:label-name, la cible de mise à jour est la ressource identifiée par l'URL de demande, et la source de mise à jour est la version choisie par l'étiquette spécifiée à partir de l'historique de version de la cible de mise à jour.

Surveillance supplémentaire :

```
<!ELEMENT update TOUTE>
  valeur de TOUTE : séquence d'éléments avec au plus un élément DAV:label-name ou DAV:version (mais pas les
                    deux).
<!ELEMENT label-name (#PCDATA)>
  valeur de PCDATA : chaîne
```

La demande PEUT inclure un en-tête Depth. Si aucun en-tête Depth n'est inclus, on suppose Depth:0. La sémantique standard de Depth s'applique, et la demande est appliquée à la collection identifiée par l'URL de demande et à tous les

membres de la collection qui satisfont à la valeur de Depth. Si un en-tête Depth est inclus et si la demande échoue sur toute ressource, la réponse DOIT être un 207 Multi-Status qui identifie toutes les ressources pour lesquelles la demande a échoué.

Préconditions supplémentaires :
(DAV:must-select-version-in-history) : Si la demande inclut un élément DAV:label-name dans le corps de demande, l'étiquette DOIT choisir une version dans l'historique de version de la ressource de version contrôlée identifiée par l'URL de demande.

(DAV:depth-update) : Si la demande inclut un en-tête Depth, la sémantique standard de Depth s'applique, et la demande est appliquée à la collection identifiée par l'URL de demande et à tous les membres de la collection qui satisfont à la valeur de Depth. La demande DOIT être appliquée à une collection avant d'être appliquée à tout membre de cette collection, car une mise à jour d'une collection de versions contrôlées pourrait changer les membres de cette collection.

Postconditions supplémentaires : (DAV:apply-request-to-labeled-version) : Si un élément DAV:label-name apparaît dans le corps de demande, le contenu et les propriétés mortes de la ressource de version contrôlée doivent avoir été mis à jour sur ceux de la version choisie par cette étiquette.

9. Caractéristique Working-Resource

La caractéristique working-resource *(ressource de travail)* donne une solution de remplacement à la caractéristique workspace *(espace de travail)* pour prendre en charge un développement en parallèle. À la différence de la caractéristique workspace, où la configuration désirée des versions et des ressources désenregistrées est conservée sur le serveur, la caractéristique working-resource conserve la configuration sur le client. Cela simplifie la mise en œuvre de serveur, mais ne permet pas à un utilisateur d'accéder à la configuration à partir de clients dans des localisations physiques différentes, comme d'un autre bureau, du domicile, ou en voyage. Une autre différence est que la caractéristique workspace isole les clients d'un changement logique qui implique de changer le nom des ressources partagées, jusqu'à ce que le changement logique soit terminé et vérifié ; avec la caractéristique working-resource, tous les clients utilisent un ensemble commun de ressources de version contrôlée partagé et chaque client voit le résultat d'un MOVE aussitôt qu'il survient.

Si un serveur prend en charge la caractéristique working-resource mais pas la caractéristique checkout-in-place, une demande CHECKOUT peut seulement être utilisée pour créer une ressource de travail, et ne peut pas être utilisée pour désenregistrer une ressource de version contrôlée. Si un serveur prend en charge la caractéristique checkout-in-place, mais pas la caractéristique working-resource, un CHECKOUT peut seulement être utilisé pour changer l'état d'une ressource de version contrôlée de enregistré à désenregistré.

9.1 Propriétés supplémentaires de Version

La caractéristique working-resource introduit les propriétés EXIGÉES suivantes pour une version.

9.1.1 DAV:checkout-fork

Cette propriété est définie au paragraphe 4.1.1.

9.1.2 DAV:checkin-fork

Cette propriété est définie au paragraphe 4.1.2.

9.2 Propriétés de ressource de travail

La caractéristique working-resource introduit les propriétés EXIGÉES suivantes pour une ressource de travail. Comme une ressource de travail est une ressource désenregistrée, elle a aussi toute propriété définie dans le présent document pour une ressource désenregistrée.

9.2.1 DAV:auto-update (protégé)

Cette propriété identifie la ressource de version contrôlée qui sera mise à jour lorsque la ressource de travail est enregistrée.

<!ELEMENT auto-update (href)>

9.2.2 DAV:checkout-fork

Cette propriété est définie au paragraphe 4.2.1.

9.2.3 DAV:checkin-fork

Cette propriété est définie au paragraphe 4.2.2.

9.3 Méthode CHECKOUT (appliquée à une version)

Une demande CHECKOUT peut être appliquée à une version pour créer une nouvelle ressource de travail. Le contenu et les propriétés mortes de la ressource de travail sont une copie de la version qui a été désenregistrée.

Surveillance : Si un corps de demande est inclus, il DOIT être un élément XML DAV:checkout.

<!ELEMENT checkout TOUTE>
valeur de TOUTE : séquence d'éléments avec au plus un élément DAV:apply-to-version et au plus un élément DAV:fork-ok.
<!ELEMENT apply-to-version VIDE>
<!ELEMENT fork-ok VIDE>

Si un corps de réponse pour une demande réussie est inclus, il DOIT être un élément XML DAV:checkout-response.

<!ELEMENT checkout-response TOUTE>

La réponse DOIT inclure un en-tête Localisation.

La réponse DOIT inclure un en-tête Cache-Control:no-cache.

Préconditions :
(DAV:checkout-of-version-with-descendant-is-forbidden) : voir au paragraphe 4.3.
(DAV:checkout-of-version-with-descendant-is-discouraged) : voir au paragraphe 4.3.
(DAV:checkout-of-checked-out-version-is-forbidden) : voir au paragraphe 4.3.
(DAV:checkout-of-checked-out-version-is-discouraged) : voir au paragraphe 4.3.

Postconditions :
(DAV:create-working-resource) *(créer une ressource de travail)* : Si l'URL de demande identifiait une version, l'en-tête Réponse de localisation DOIT contenir l'URL d'une nouvelle ressource de travail. La propriété DAV:checked-out de la nouvelle ressource de travail DOIT identifier la version qui a été désenregistrée. Le contenu et les propriétés mortes de la ressource de travail DOIVENT être des copies du contenu et des propriétés mortes de la version DAV:checked-out. La propriété DAV:predecessor-set de la ressource de travail DOIT être initialisée comme étant la version identifiée par l'URL de demande. La propriété DAV:auto-update de la ressource de travail NE DOIT PAS exister.

(DAV:create-working-resource-from-checked-in-version) *(créer une ressource de travail à partir de la version enregistrée)* : Si l'URL de demande identifiait une ressource de version contrôlée, et si DAV:apply-to-version est spécifié dans le corps de demande, le CHECKOUT est appliqué à la version DAV:checked-in de la ressource de version contrôlée, et non à la ressource de version contrôlée elle-même. Une nouvelle ressource de travail est créée et la ressource de version contrôlée reste enregistrée. La propriété DAV:auto-update de la ressource de travail DOIT identifier la ressource de version contrôlée.

9.3.1 Exemple - CHECKOUT d'une version

>>REQUEST
CHECKOUT /his/12/ver/V3 HTTP/1.1
Host: repo.webdav.org
Content-Length: 0

>>RESPONSE
HTTP/1.1 201 Created
Location: http://repo.webdav.org/wr/157
Cache-Control: no-cache

Dans cet exemple, la version identifiée par http://repo.webdav.org/his/12/ver/V3 est désenregistrée, et la nouvelle ressource de travail est située à http://repo.webdav.org/wr/157.

9.4 Méthode CHECKIN (appliquée à une ressource de travail)

Une demande CHECKIN peut être appliquée à une ressource de travail pour produire une nouvelle version dont le contenu et les propriétés mortes sont une copie de celles de la ressource de travail. Si la propriété DAV:auto-update de la ressource de travail a été établie parce que la ressource de travail a été créée en appliquant un CHECKOUT avec le fanion DAV:apply-to-version à une ressource de version contrôlée, la demande CHECKIN va aussi mettre à jour le contenu et les propriétés mortes de cette ressource de version contrôlée sur celles de la nouvelle version.

Surveillance : Si un corps de demande est inclus, il DOIT être un élément XML DAV:checkin.

```
<!ELEMENT checkin TOUTE>
  valeur de TOUTE : séquence d'éléments avec au plus un élément DAV:fork-ok.
<!ELEMENT fork-ok VIDE>
```

Si un corps de réponse pour une demande réussie est inclus, il DOIT être un élément XML DAV:checkin-response.

```
<!ELEMENT checkin-response TOUTE>
```

La réponse DOIT inclure un en-tête Cache-Control:no-cache.

Préconditions :
(DAV:must-be-checked-out) : voir au paragraphe 4.4.
(DAV:version-history-is-tree) : voir au paragraphe 4.4.
(DAV:checkin-fork-forbidden) : voir au paragraphe 4.4.
(DAV:checkin-fork-discouraged) : voir au paragraphe 4.4.
(DAV:no-overwrite-by-auto-update) : Si la propriété DAV:auto-update pour la ressource désenregistrée identifie une ressource de version contrôlée, au moins une des versions identifiées par la propriété DAV:predecessor-set de la ressource désenregistrée DOIT identifier une version qui soit la même que, ou un descendant de, la version identifiée par la propriété DAV:checked-in de cette ressource de version contrôlée.

Postconditions :
(DAV:create-version) : voir au paragraphe 4.4.
(DAV:initialize-version-content-and-propriétés) : voir au paragraphe 4.4.
(DAV:auto-update) : Si la propriété DAV:auto-update de la ressource désenregistrée identifiait une ressource de version contrôlée, une demande UPDATE avec la nouvelle version DOIT avoir été appliquée à cette ressource de version contrôlée.

(DAV:delete-working-ressource) : Si l'URL de demande identifie une ressource de travail et si DAV:keep-checked-out n'est pas spécifié dans le corps de demande, la ressource de travail est supprimée.

9.4.1 Exemple - CHECKIN d'une ressource de travail

>>REQUEST
CHECKIN /wr/157 HTTP/1.1
Host: repo.webdav.org
Content-Length: 0

>>RESPONSE
HTTP/1.1 201 Created
Location: http://repo.webdav.org/his/23/ver/15
Cache-Control: no-cache

Dans cet exemple, la ressource de travail /wr/157 s'est enregistrée, et une nouvelle version est créée à http://repo.webdav.org/his/23/ver/15.

9.5 Sémantique supplémentaire de OPTIONS

Si le serveur prend en charge la caractéristique working-resource, il DOIT inclure "working-resource" comme champ dans l'en-tête de réponse DAV d'une demande OPTIONS sur toute ressource qui prend en charge des propriétés, rapports, ou méthodes de gestion de version.

9.6 Sémantique supplémentaire de COPY

Postconditions supplémentaires : (DAV:copy-creates-new-ressource) : Le résultat de la copie d'une ressource de travail est une nouvelle ressource de version non contrôlée à la destination de la COPY. La nouvelle ressource PEUT automatiquement être mise sous un contrôle de version, mais la ressource de version contrôlée résultante DOIT être associée à un nouvel historique de version créé pour cette nouvelle ressource de version contrôlée.

9.7 Sémantique supplémentaire de MOVE

Préconditions supplémentaires : (DAV:cannot-rename-working-ressource) : Si l'URL de demande identifie une ressource de travail, la demande DOIT échouer.

Postconditions supplémentaires : (DAV:update-auto-update) : Si l'URL de demande identifiait une ressource de version contrôlée, toutes les propriétés DAV:auto-update qui identifiaient cette ressource de version contrôlée DOIVENT avoir été mises à jour pour contenir la nouvelle localisation de cette ressource de version contrôlée.

10. Caractéristiques avancées de gestion de version

La gestion avancée de version s'adresse aux problèmes de développement parallèle et de gestion de configuration de plusieurs ensembles de ressources qui ont des relations entre eux. Traditionnellement, des artifices de développement logiciel, incluant des exigences, des documents de conception, du code, et des cas d'essai, ont été la cible de la gestion de configuration. Les sites de la Toile, comprenant plusieurs ressources entrelacées (HTML, graphiques, son, CGI, et autres) sont une autre classe d'artifices d'information complexes qui bénéficient de l'application de gestion de configuration. Les capacités avancées de gestion de version pour coordonner les changements concurrents fournissent l'infrastructure pour une gestion efficace et contrôlée de grands sites évolutifs de la Toile.

10.1 Paquetages avancés de gestion de version

Bien qu'un serveur PUISSE prendre en charge toute combinaison de caractéristiques avancées de gestion de version, afin de minimiser la complexité d'un client de gestion de version WebDAV avancé, un serveur WebDAV de gestion avancée de version DEVRAIT prendre en charge un des paquetages suivants :
- Paquetage Advanced-Server-Workspace : paquetage basic-server-workspace plus toutes les caractéristiques avancées.
- Paquetage Advanced-Client-Workspace : paquetage basic-client-workspace plus toutes les caractéristiques avancées.

Le paquetage advanced-server-workspace prend en charge les capacités avancées de gestion de version pour un client sans état persistant. Le paquetage advanced-client-workspace prend en charge les capacités avancées de gestion de version pour un client qui conserve l'état de configuration sur le client. Un serveur qui prend en charge les deux paquetages d'espace de travail avancé va interopérer avec tous les clients de gestion de version.

10.2 Termes de gestion de version avancée

Les termes supplémentaires suivants sont utilisés par les caractéristiques de gestion de version avancées.

Collection
Une "collection" est une ressource dont l'état consiste non seulement en contenu et propriétés, mais aussi en "liens" désignés, où un lien identifie ce que la RFC 2518 appelle un "membre interne" de la collection. Noter que un lien n'est pas une ressource, mais fait plutôt partie de l'état d'une collection qui définit une transposition d'un nom de lien (un segment d'URL) en une ressource (un membre interne de la collection).

Ressource de version de collection
Une "ressource de version de collection", ou simplement une "version de collection", capture les propriétés mortes d'une collection de versions contrôlées, ainsi que les noms de ses liens de version contrôlée (voir la Section 14). Un lien de version contrôlée est un lien avec une ressource de version contrôlée. Si la caractéristique checkout-in-place est prise en charge, une version de collection peut être créée en désenregistrant puis en enregistrant dans une collection de versions contrôlées. Si la caractéristique de ressource de travail est prise en charge, une version de collection peut être créée en désenregistrant une version de collection (pour créer une "collection de travail") puis en enregistrant la collection de travail.

Configuration
Une "configuration" est un ensemble de ressources qui consiste en une collection racine et tous les membres (pas seulement les membres internes) de cette collection racine qui ne sont pas des membres d'une autre configuration. La collection racine est appelée "racine de configuration", et les membres de cet ensemble sont appelés les "membres de la configuration". Noter qu'une collection (qui est une seule ressource) est très différente d'une configuration (qui est un ensemble de ressources).

Ressource de ligne de base
Une "ressource de ligne de base", ou simplement "ligne de base", d'une collection est une version de la configuration qui a sa racine à cette collection (voir la Section 12). En particulier, une ligne de base capture la version DAV:checked-in de chaque membre de version contrôlée de cette configuration. Noter qu'une version de collection (qui capture l'état d'une seule ressource) est très différente d'une ligne de base de collection (qui capture l'état d'un ensemble de ressources).

Collection de ligne de base contrôlée
Une "collection de ligne de base contrôlée" est une collection à partir de laquelle des lignes de base peuvent être créées (voir la Section 12).

Ressource de configuration de version contrôlée
Une "ressource de configuration de version contrôlée", ou simplement "configuration de version contrôlée", est une sorte de ressource de version contrôlée particulière qui est associée à une collection de ligne de base contrôlée, et est utilisée pour créer et accéder à des lignes de base de cette collection (voir la Section 12). Lorsque une collection est à la fois à version contrôlée et à ligne de base contrôlée, un client peut créer une nouvelle version de la collection en désenregistrant et en enregistrant cette collection, et il peut créer une nouvelle ligne de base de cette collection en désenregistrant et en enregistrant dans la configuration de version contrôlée de cette collection.

Ressource d'activité
Une "ressource d'activité", ou simplement "activité", est une ressource qui choisit un ensemble de versions qui correspond à un seul changement logique, où les versions choisies à partir d'un certain historique de version forment une seule ligne de descente à travers cet historique de version (voir la Section 13).

11. Caractéristique Merge

Lorsque un usager veut accepter les changements (nouvelles versions) créées par quelqu'un d'autre, il est important de ne pas juste mettre à jour les ressources de version contrôlée dans l'espace de travail de l'usager avec ces nouvelles versions, car il pourrait en résulter une "exclusion" des changements que l'usager a fait à ces ressources de version contrôlée. Les versions créées dans un autre espace de travail devraient plutôt être "fusionnées" avec les ressources de version contrôlée de l'usager.

L'historique de version d'une ressource de version contrôlée fournit les informations nécessaires pour déterminer le résultat de la fusion. En particulier, la fusion devrait choisir quelle version est la dernière dans la ligne de descente à partir de la version racine. Au cas où les versions à fusionner sont sur des lignes de descente différentes (aucune des versions n'est un descendant de l'autre) aucune version ne devrait être choisie, mais une nouvelle version devrait plutôt être créée contenant la fusion logique du contenu et des propriétés mortes de ces versions. La demande MERGE peut être utilisée pour désenregistrer chaque ressource de version contrôlée qui exige une telle fusion, et établir la propriété DAV:merge-set de chaque ressource désenregistrée pour identifier la version à fusionner. L'usager est responsable de la modification du contenu et des propriétés mortes de la ressource désenregistrée afin qu'elle représente la fusion logique de cette version, et d'ajouter ensuite cette version au DAV:predecessor-set de la ressource désenregistrée.

Si le serveur est capable d'effectuer automatiquement la fusion, il PEUT mettre à jour le contenu, les propriétés mortes, et le DAV:predecessor-set de la ressource désenregistrée elle-même. Avant d'enregistrer la ressource fusionnée automatiquement, l'usager est responsable de la vérification de la correction de la fusion automatique.

11.1 Propriétés supplémentaires de ressource désenregistrée

La caractéristique merge introduit les propriétés EXIGÉES suivantes pour une ressource désenregistrée.

11.1.1 DAV:merge-set

Cette propriété identifie chaque version qui est à fusionner dans cette ressource désenregistrée.

```
<!ELEMENT merge-set (href*)>
```

11.1.2 DAV:auto-merge-set

Cette propriété identifie chaque version que le serveur a fusionné dans cette ressource désenregistrée. Le client devrait confirmer que la fusion a été effectuée correctement avant de déplacer un URL du DAV:auto-merge-set au DAV:predecessor-set d'une ressource désenregistrée.

```
<!ELEMENT auto-merge-set (href*)>
```

11.2 Méthode MERGE

La méthode MERGE effectue la fusion logique d'une version spécifiée (la "source de fusion") dans une ressource de version contrôlée spécifiée (la "cible de fusion"). Si la source de fusion n'est ni un ancêtre ni un descendant de la version DAV:checked-in ou DAV:checked-out de la cible de fusion, la méthode MERGE désenregistre la cible de fusion (si elle n'est pas déjà désenregistrée) et ajoute l'URL de la source de fusion avec le DAV:merge-set de la cible de fusion. C'est alors la responsabilité du client de mettre à jour le contenu et les propriétés mortes de la cible de fusion désenregistrée afin qu'elle reflète la fusion logique de la source de fusion dans l'état actuel de la cible de fusion. Le client indique qu'il a terminé la mise à jour de la cible de fusion en supprimant l'URL de source de fusion à partir du DAV:merge-set de la cible de fusion désenregistrée, et en l'ajoutant au DAV:predecessor-set. Comme vérification pour un client qui oublie de terminer une fusion, le serveur DOIT faire échouer une tentative de CHECKIN d'une ressource de version contrôlée avec un DAV:merge-set non vide.

Lorsque un serveur a la capacité de mettre à jour automatiquement le contenu et les propriétés mortes de la cible de fusion pour refléter la fusion logique de la source de fusion, il peut le faire sauf si le DAV:no-auto-merge est spécifié dans le corps de demande MERGE. Afin de notifier au client qu'une source de fusion a été automatiquement fusionnée, la demande MERGE DOIT ajouter l'URL de la source auto-fusionnée à la propriété DAV:auto-merge-set de la cible de fusion, et non à la propriété DAV:merge-set. Le client indique qu'il a vérifié que l'auto-fusion est valide, en supprimant l'URL de source de fusion du DAV:auto-merge-set, et en l'ajoutant au DAV:predecessor-set.

Plusieurs sources de fusion peuvent être spécifiées dans une seule demande MERGE. L'ensemble des sources de fusion d'une demande MERGE est déterminé à partir de l'élément DAV:source du corps de demande MERGE comme suit :

- Si DAV:source identifie une version, cette version est une source de fusion.
- Si DAV:source identifie une ressource de version contrôlée, la version DAV:checked-in de cette ressource de version contrôlée est une source de fusion.
- Si DAV:source identifie une collection, la version DAV:checked-in de chaque ressource de version contrôlée qui est membre de cette collection est une source de fusion.

L'URL de demande identifie l'ensemble de cibles de fusion possibles. Si l'URL de demande identifie une collection, tout membre de la configuration qui a sa racine à l'URL de demande est une cible de fusion possible. La cible de fusion d'une source de fusion particulière est la ressource de version contrôlée, ou la ressource désenregistrée, dont la version DAV:checked-in ou DAV:checked-out est du même historique de version que la source de fusion. Si une source de fusion n'a pas de cible de fusion, cette source de fusion est ignorée.

La réponse MERGE identifie les ressources qu'un client doit modifier pour achever la fusion. Elle identifie aussi les ressources modifiées par la demande, de sorte qu'un client peut efficacement mettre à jour tout état en antémémoire qu'il conserve.

Surveillance : Le corps de demande DOIT être un élément DAV:merge.

L'ensemble de sources de fusion est déterminé par l'élément DAV:source dans le corps de demande.

<!ELEMENT merge TOUTE
valeur de TOUTE : séquence d'éléments avec un élément DAV:source, au plus un élément DAV:no-auto-merge, au plus un élément DAV:no-checkout, au plus un élément DAV:prop, et tout ensemble légal d'éléments qui peuvent survenir dans un élément DAV:checkout.
<!ELEMENT source (href+)>
<!ELEMENT no-auto-merge VIDE>
<!ELEMENT no-checkout VIDE>
prop : voir la RFC 2518, paragraphe 12.11

La réponse pour une demande réussie DOIT être un 207 Multi-Status, où l'élément XML DAV:multistatus dans le corps de réponse identifie toutes les ressources qui ont été modifiées par la demande.
Multistatus : voir la RFC 2518, paragraphe 12.9

La réponse à une demande réussie DOIT inclure un en-tête Localisation contenant l'URL pour la nouvelle version créée par l'enregistrement.

La réponse DOIT inclure un en-tête Cache-Control:no-cache.

Préconditions : (DAV:cannot-merge-checked-out-ressource) : L'élément DAV:source NE DOIT PAS identifier une ressource désenregistrée. Si l'élément DAV:source identifie une collection, la collection NE DOIT PAS avoir un membre qui soit une ressource désenregistrée.

(DAV:checkout-not-allowed) : Si DAV:no-checkout est spécifié dans le corps de demande, il DOIT être possible d'effectuer la fusion sans désenregistrer aucune cible de fusion.

Toutes les préconditions de l'opération CHECKOUT s'appliquent aux caractéristiques effectués par la demande.

Postconditions :
(DAV:ancestor-version) : Si une cible de fusion est une ressource de version contrôlée ou une ressource désenregistrée dont la version DAV:checked-in ou DAV:checked-out est la source de fusion ou est un descendant de la source de fusion, la cible de fusion NE DOIT PAS avoir été modifiée par le MERGE.

(DAV:descendant-version) : Si la cible de fusion était une ressource de version contrôlée désenregistrée dont la version DAV:checked-in était un ancêtre de la source de fusion, une opération UPDATE DOIT avoir été appliquée à la cible de fusion pour régler son contenu et ses propriétés mortes à ceux de la source de fusion. Si la méthode UPDATE n'est pas acceptée, la cible de fusion DOIT avoir été désenregistrée, le contenu et les propriétés mortes de la cible de fusion DOIVENT avoir été réglés à ceux de la source de fusion, et la source de fusion DOIT avoir été ajoutée au DAV:auto-merge-set de la cible de fusion. La cible de fusion DOIT apparaître dans un élément XML DAV:response dans le corps de réponse.

(DAV:checked-out-for-merge) : Si la cible de fusion était une ressource de version contrôlée enregistrée dont la version DAV:checked-in n'était ni un descendant ni un ancêtre de la source de fusion, un CHECKOUT DOIT avoir été appliqué à la cible de fusion. Tous les éléments XML dans l'élément XML DAV:merge qui pourraient apparaître dans un élément XML DAV:checkout DOIVENT avoir été utilisés comme arguments de la demande CHECKOUT. La cible de fusion DOIT apparaître dans un élément XML DAV:response dans le corps de réponse.

(DAV:update-merge-set) : Si la version DAV:checked-out de la cible de fusion n'est ni égale à la source de fusion ni à un de ses descendants, la source de fusion DOIT être ajoutée au DAV:merge-set ou au DAV:auto-merge-set de la cible de fusion. La cible de fusion DOIT apparaître dans un élément XML DAV:response dans le corps de réponse.

Si une source de fusion a été ajoutée au DAV:auto-merge-set, le contenu et les propriétés mortes de la cible de fusion DOIVENT avoir été modifiés par le serveur pour refléter le résultat d'une fusion logique de la source de fusion et de la cible de fusion. Si une source de fusion a été ajoutée au DAV:merge-set, le contenu et les propriétés mortes de la cible de fusion NE DOIVENT PAS avoir été modifiés par le serveur. Si DAV:no-auto-merge est spécifié dans le corps de demande, la source de fusion NE DOIT PAS avoir été ajoutée au DAV:auto-merge-set.

(DAV:report-properties) : Si DAV:prop est spécifié dans le corps de demande, les propriétés spécifiées dans l'élément DAV:prop DOIVENT être rapportées dans les éléments DAV:response dans le corps de réponse.

11.2.1 Exemple - MERGE

>>REQUEST

```
MERGE /ws/public HTTP/1.1
Host: www.webdav.org
Content-type: text/xml; charset="utf-8"
Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:merge xmlns:D="DAV:">
     <D:source>
       <D:href>http://www.webdav.org/ws/dev/sally</D:href>
     </D:source>
   </D:merge>

>>RESPONSE
HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
Cache-Control: no-cache

   <?xml version="1.0" encoding="utf-8" ?>
   <D:multistatus xmlns:D="DAV:">
     <D:response>
  <D:href>http://www.webdav.org/ws/public/src/parse.c</D:href>
  <D:status>HTTP/1.1 200 OK</D:status>
     </D:response>
     <D:response>
  <D:href>http://www.webdav.org/ws/public/doc/parse.html</D:href>
  <D:status>HTTP/1.1 200 OK</D:status>
     </D:response>
   </D:multistatus>
```

Dans cet exemple, les versions DAV:checked-in provenant de l'espace de travail http://www.webdav.org/ws/dev/sally sont fusionnées dans les ressources de version contrôlée dans l'espace de travail http://www.webdav.org/ws/public. Les ressources /ws/public/src/parse.c et /ws/public/doc/parse.html ont été modifiées par la demande.

11.3 Rapport DAV:merge-preview

Une prévision de fusion décrit les changements qui résulteraient si les versions spécifiées par l'élément DAV:source dans le corps de demande devaient être fusionnées dans la ressource identifiée par l'URL de demande (en général, une collection).

Surveillance : Le corps de demande DOIT être un élément XML DAV:merge-preview.

```
<!ELEMENT merge-preview (source)>
<!ELEMENT source (href)>
```

Le corps de réponse pour une demande réussie DOIT être un élément XML DAV:merge-preview-report.

```
<!ELEMENT merge-preview-report (update-preview | conflict-preview | ignore-preview)*>
```

Un élément DAV:update-preview identifie une cible de fusion dont la propriété DAV:checked-in changerait par suite du MERGE, et identifie la source de fusion pour cette cible de fusion.

```
<!ELEMENT update-preview (target, version)>
<!ELEMENT target (href)>
<!ELEMENT version (href)>
```

Un élément DAV:conflict-preview identifie une cible de fusion qui exige une fusion.

```
<!ELEMENT conflict-preview (cible, ancêtre commun, version)>
```

Un élément DAV:common-ancestor identifie la version qui est un ancêtre commun de la source de fusion et de la version DAV:checked-in ou DAV:checked-out de la cible de fusion.

```
<!ELEMENT ancêtre commun (href)>
```

Un élément DAV:ignore-preview identifie une version qui n'a pas de cible de fusion et sera donc ignorée par la fusion.

```
<!ELEMENT ignore-preview (version)>
```

11.3.1 Exemple – rapport DAV:merge-preview

```
>>REQUEST
REPORT /ws/public HTTP/1.1
Host: www.webdav.org
Content-type: text/xml; charset="utf-8"
Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:merge-preview xmlns:D="DAV:">
    <D:source>
     <D:href>http://www.webdav.org/ws/dev/fred</D:href>
    </D:source>
   </D:merge-preview>

>>RESPONSE
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:merge-preview-report xmlns:D="DAV:">
    <D:conflict-preview>
     <D:target>
      <D:href>http://www.webdav.org/ws/public/foo.html</D:href>
     </D:target>
     <D:common-ancestor>
      <D:href>http://repo.webdav.org/his/23/ver/18</D:href>
     </D:common-ancestor>
     <D:version>
      <D:href>http://repo.webdav.org/his/23/ver/42</D:href>
     </D:version>
    </D:conflict-preview>
    <D:update-preview>
     <D:target>
      <D:href>http://www.webdav.org/ws/public/bar.html</D:href>
     </D:target>
     <D:version>
      <D:href>http://www.repo/his/42/ver/3</D:href>
     </D:version>
    </D:update-preview>
   </D:merge-preview-report>
```

Dans cet exemple, le rapport de prévision de fusion indique que la version /his/23/ver/42 serait fusionnée dans /ws/public/foo.html, et que la version /his/42/ver/3 mettrait à jour /ws/public/bar.html si l'espace de travail http://www.webdav.org/ws/dev/fred était fusionné dans l'espace de travail http://www.webdav.org/ws/public.

11.4 Sémantique supplémentaire de OPTIONS

Si le serveur accepte la caractéristique merge, il DOIT inclure "merge" comme champ dans l'en-tête de réponse DAV provenant d'une demande OPTIONS sur toute ressource qui prend en charge des propriétés, rapports, ou méthodes de gestion de version.

11.5 Sémantique supplémentaire de DELETE

Postcondition supplémentaire : (DAV:delete-version-reference) : Si une version est supprimée, toute référence à cette version dans une propriété DAV:merge-set ou DAV:auto-merge-set DOIT être retirée.

11.6 Sémantique supplémentaire de CHECKIN

Précondition supplémentaire : (DAV:merge-must-be-complete) : Le DAV:merge-set et le DAV:auto-merge-set de la ressource désenregistrée DOIVENT être vides ou ne pas exister.

12. Caractéristiques de base

Une configuration est un ensemble de ressources qui consiste en une collection racine et tous les membres de cette collection racine sauf les ressources qui sont membres d'une autre configuration. Une configuration qui contient un grand nombre de ressources peut consommer une grande quantité d'espace sur un serveur. Cela peut rendre excessivement coûteux de se rappeler l'état d'une configuration existante en créant une copie Depth:infinity de sa collection racine.

Une ligne de base *(baseline)* est une ressource de version qui capture l'état de chaque version contrôlée membre d'une configuration. Un historique de ligne de base est un historique de version dont les versions sont des lignes de base. De nouvelles lignes de base sont créées en désenregistrant puis en enregistrant une ressource de version contrôlée d'une espèce spéciale appelée une configuration de version contrôlée.

Une collection qui est sous contrôle de ligne de base est appelée une collection à contrôle de ligne de base. Pour permettre une mise en œuvre efficace de ligne de base, l'état de ligne de base d'une collection est limité à être un ensemble de versions et leurs noms par rapport à la collection, et les opérations sur une ligne de base sont limitées à la création d'une ligne de base à partir d'une collection, et à la restauration ou la fusion de la ligne de base dans une collection. Un serveur PEUT automatiquement mettre une collection sous contrôle de ligne de base lorsque elle est créée, ou un client peut utiliser la méthode BASELINE-CONTROL pour mettre une collection spécifiée sous contrôle de ligne de base.

Lorsque une configuration devient importante, il est souvent utile de la casser en un ensemble de plus petites configurations qui forment les "composants" logiques de cette configuration. Afin de capturer le fait qu'une ligne de base d'une configuration est logiquement étendue par une ligne de base de configuration composante, la ligne de base de configuration composante est capturée comme "sous ligne de base" de la ligne de base.

La collection racine d'une configuration n'est contrainte dans ses relations à la collection racine d'aucun de ses composants. En particulier, la collection racine d'une configuration peut avoir un membre qui est la collection racine d'un de ses composants (par exemple, la configuration /sys/x peut avoir un composant /sys/x/foo), peut être un membre de la collection racine d'un de ses composants (par exemple, la configuration /sys/y/z peut avoir un composant /sys/y), ou aucun des deux (par exemple, la configuration /sys/x peut avoir un composant /comp/bar).

12.1 Propriétés de configuration de version contrôlée

Comme une configuration de version contrôlée est une ressource de version contrôlée, elle a toutes les propriétés d'une ressource de version contrôlée. De plus, la caractéristique baseline introduit la propriété EXIGÉE suivante pour une configuration de version contrôlée.

12.1.1 DAV:baseline-controlled-collection (protégé)

Cette propriété identifie la collection qui contient les ressources de version contrôlée dont les versions DAV:checked-in sont retracées par cette configuration de version contrôlée. Le DAV:version-controlled-configuration de la DAV:baseline-controlled-collection d'une configuration de version contrôlée DOIT identifier cette configuration de version contrôlée.

```
<!ELEMENT baseline-controlled-collection (href)>
```

12.2 Propriétés de configuration désenregistrée

Comme une configuration désenregistrée est une ressource désenregistrée, elle a toutes les propriétés d'une ressource désenregistrée. De plus, la caractéristique baseline introduit la propriété EXIGÉE suivante pour une configuration désenregistrée.

12.2.1 DAV:subbaseline-set

Cette propriété détermine la propriété DAV:subbaseline-set de la ligne de base qui résulte de l'enregistrement de cette

ressource.

Un serveur PEUT rejeter les tentatives de modification du DAV:subbaseline-set d'une configuration désenregistrée.

<!ELEMENT subbaseline-set (href*)>

12.3 Propriétés de base

Le DAV:resourcetype d'une ligne de base DOIT être DAV:baseline. Comme une ligne de base est une ressource de version, elle a toutes les propriétés d'une ressource de version. De plus, la caractéristique baseline introduit les propriétés EXIGÉES suivantes pour une ligne de base.

12.3.1 DAV:baseline-collection (protégé)

Cette propriété contient un URL défini par le serveur pour une collection, où chaque membre de cette collection DOIT soit être une ressource de version contrôlée avec la même version DAV:checked-in et le même nom relatif qu'un membre de version contrôlée de la collection de ligne de base contrôlée au moment de la création de la ligne de base, soit être une collection nécessaire pour fournir le nom relatif pour une ressource de version contrôlée.

<!ELEMENT baseline-collection (href)>

12.3.2 DAV:subbaseline-set (protégé)

Les URL dans la propriété DAV:subbaseline-set DOIVENT identifier un ensemble d'autres lignes de base. Les sous lignes de base d'une ligne de base sont les lignes de base identifiées par son DAV:subbaseline-set et toutes les sous lignes de base et les lignes de base identifiées par son DAV:subbaseline-set.

<!ELEMENT subbaseline-set (href*)>

12.4 Propriétés de ressource supplémentaires

La caractéristique baseline introduit la propriété EXIGÉE suivante pour une ressource.

12.4.1 DAV:version-controlled-configuration (calculé)

Si la ressource est un membre d'une configuration de version contrôlée (c'est-à-dire, la ressource est une collection sous contrôle de ligne de base ou est un membre d'une collection sous contrôle de ligne de base) cette propriété identifie cette configuration de version contrôlée.

<!ELEMENT version-controlled-configuration (href)>

12.5 Propriétés supplémentaires d'espace de travail

La caractéristique baseline introduit la propriété EXIGÉE suivante pour un espace de travail.

12.5.1 DAV:baseline-controlled-collection-set (calculé)

Cette propriété identifie chaque membre de l'espace de travail qui est une collection sous contrôle de ligne de base (ainsi que l'espace de travail lui-même, si il est sous contrôle de ligne de base).

<!ELEMENT baseline-controlled-collection-set (href*)>

12.6 Méthode BASELINE-CONTROL

Une collection peut être placée sous contrôle de ligne de base avec une demande BASELINE-CONTROL. Lorsqu'une collection est placée sous contrôle de ligne de base, la propriété DAV:version-controlled-configuration de la collection est réglée à identifier une nouvelle configuration de version contrôlée. Cette configuration de version contrôlée peut être désenregistrée et ensuite enregistrée pour créer une nouvelle ligne de base pour cette collection.

Si une ligne de base est spécifiée dans le corps de demande, la version DAV:checked-in de la nouvelle configuration de version contrôlée sera cette ligne de base, et la collection est initialisée pour contenir des membres de version contrôlée dont les versions DAV:checked-in et les noms relatifs sont déterminés par la ligne de base spécifiée.

Si aucune ligne de base n'est spécifiée, un nouvel historique de ligne de base est créé qui contient une ligne de base qui capture l'état des membres de version contrôlée de la collection, et la version DAV:checked-in de la configuration de version contrôlée sera cette ligne de base.

Surveillance : Si un corps de demande est inclus, il DOIT être un élément XML DAV:baseline-control.

```
<!ELEMENT baseline-control TOUTE>
  valeur de TOUTE : séquence d'éléments avec au plus un élément DAV:baseline.
<!ELEMENT baseline (href)>
```

Si un corps de réponse pour une demande réussie est inclus, il DOIT être un élément XML DAV:baseline-control-response.

```
<!ELEMENT baseline-control-response ANY>
```

La réponse DOIT inclure un en-tête Cache-Control:no-cache.

Préconditions : (DAV:version-controlled-configuration-must-not-exist) : La propriété DAV:version-controlled-configuration de la collection identifiée par l'URL de demande DOIT ne pas exister.
(DAV:must-be-baseline) : Le DAV:href de l'élément DAV:baseline dans le corps de demande DOIT identifier une ligne de base.
(DAV:must-have-no-version-controlled-members) : Si un élément DAV:baseline est spécifié dans le corps de demande, la collection identifiée par l'URL de demande DOIT n'avoir aucun membre de version contrôlée.
(DAV:one-baseline-controlled-collection-per-history-per-workspace) : Si l'URL de demande identifie un espace de travail ou un membre d'un espace de travail, et si une ligne de base est spécifiée dans un élément DAV:baseline dans le corps de demande, il NE DOIT PAS alors y avoir une autre collection dans cet espace de travail dont la propriété DAV:version-controlled-configuration identifie une configuration de version contrôlée pour l'historique de ligne de base de cette ligne de base.

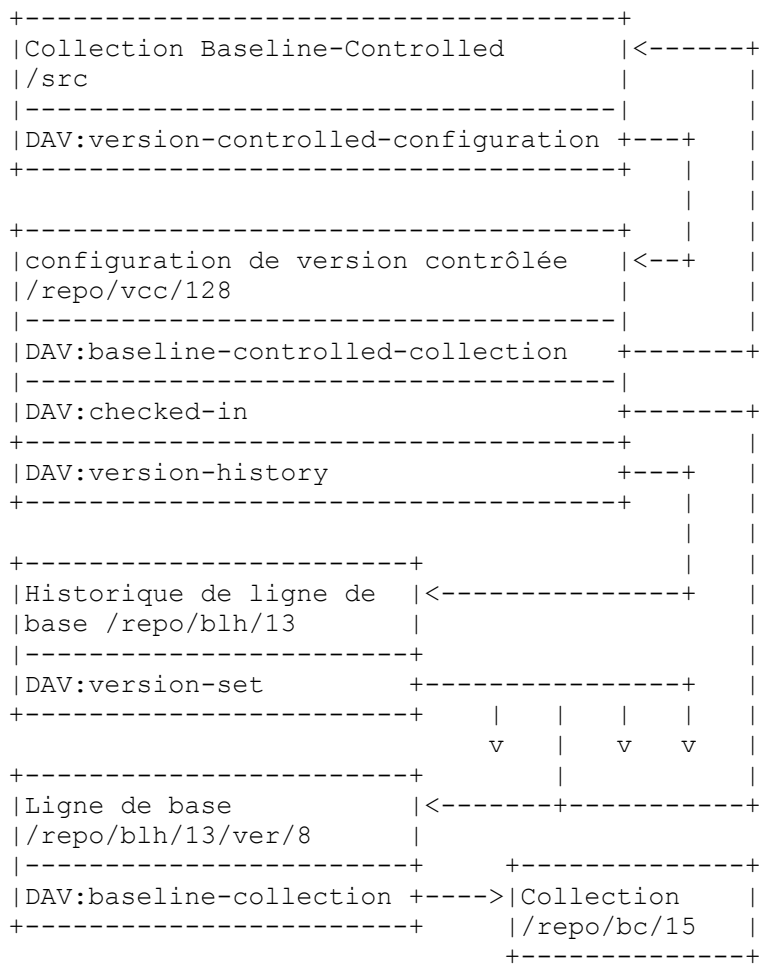
Postconditions :
(DAV:create-version-controlled-configuration) : Une nouvelle configuration de version est créée, dont la propriété DAV:baseline-controlled-collection identifie la collection.
(DAV:reference-version-controlled-configuration) : La DAV:version-controlled-configuration de la collection identifie la nouvelle configuration de version contrôlée.
(DAV:select-existing-baseline) : Si le corps de demande spécifie une ligne de base, la propriété DAV:checked-in de la nouvelle configuration de version contrôlée DOIT avoir été réglée de façon à identifier cette ligne de base. Un membre de version contrôlée de la collection sera créé pour chaque version dans la ligne de base, où le membre de version contrôlée aura le contenu et les propriétés mortes de cette version, et aura le même nom par rapport à la collection qu'avait la ressource de version contrôlée correspondante lors de la création de la ligne de base. Toutes les collections incorporées qui sont nécessaires pour fournir le nom approprié pour un membre de version contrôlée seront créées.
(DAV:create-new-baseline) : Si aucune ligne de base n'est spécifiée dans le corps de demande, la demande DOIT avoir créé un nouvel historique de ligne de base à l'URL défini par le serveur, et DOIT avoir créé une nouvelle ligne de base dans cet historique de ligne de base. La DAV:baseline-collection de la nouvelle ligne de base DOIT identifier une collection dont les membres ont le même nom relatif et la même version DAV:checked-in que les membres de la version contrôlée de la collection de demande. La propriété DAV:checked-in de la nouvelle configuration de version contrôlée DOIT identifier la nouvelle ligne de base.

12.6.1 Exemple - BASELINE-CONTROL

```
>>REQUEST
BASELINE-CONTROL /src HTTP/1.1
Host: www.webdav.org
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx
```

```
   <?xml version="1.0" encoding="utf-8" ?>
   <D:baseline-control xmlns:D="DAV:">
     <D:href>http://www.webdav.org/repo/blh/13/ver/8</D:href>
   </D:baseline-control>
>>RESPONSE
HTTP/1.1 200 OK
Cache-Control: no-cache
Content-Length: 0
```

Dans cet exemple, la collection /src est placée sous contrôle de ligne de base, et est remplie avec les membres d'une ligne de base existante. Une nouvelle configuration de version contrôlée (/repo/vcc/128) est créée et est associée à /src, et /src est initialisé avec les membres de version contrôlée dont les versions DAV:checked-in sont celles choisies par la DAV:baseline-collection (/repo/bc/15) de la ligne de base spécifiée (/repo/blh/13/ver/8). Le diagramme qui suit illustre l'état résultant chez le serveur.



Afin de créer de nouvelles lignes de base de /src, /repo/vcc/128 peut être désenregistré, de nouvelles versions peuvent être créées ou choisies par les membres de version contrôlée de /src, et ensuite /repo/vcc/128 peut être enregistré pour capturer l'état actuel de ces membres de version contrôlée.

12.7 Rapport DAV:compare-baseline

Un rapport DAV:compare-baseline contient les différences entre la ligne de base identifiée par l'URL de demande (la "ligne de base de demande") et la ligne de base spécifiée dans le corps de demande (la "ligne de base de comparaison").

Surveillance : Le corps de demande DOIT être un élément XML DAV:compare-baseline.

```
<!ELEMENT compare-baseline (href)>
```

Le corps de réponse pour une demande réussie DOIT être un élément XML DAV:compare-baseline-report.

<!ELEMENT compare-baseline-report (added-version | deleted-version | changed-version)*>

Un élément DAV:added-version identifie une version qui est la version DAV:checked-in d'un membre de la DAV:baseline-collection de la ligne de base de comparaison, mais aucune version dans l'historique de version de cette version n'est la version DAV:checked-in d'un membre de la DAV:baseline-collection de la ligne de base de demande.

<!ELEMENT added-version (href)>

Un élément DAV:deleted-version identifie une version qui est la version DAV:checked-in d'un membre de la DAV:baseline-collection de la ligne de base de demande, mais aucune version dans l'historique de version de cette version n'est la version DAV:checked-in d'un membre de la DAV:baseline-collection de la ligne de base de comparaison.

<!ELEMENT deleted-version (href)>

Un élément DAV:changed-version identifie deux versions différentes provenant du même historique de version qui sont respectivement la version DAV:checked-in de la DAV:baseline-collection de la ligne de base de demande et la ligne de base de comparaison.

<!ELEMENT changed-version (href, href)>

Préconditions :
(DAV:must-be-baseline) : Le DAV:href dans le corps de demande DOIT identifier une ligne de base.

(DAV:baselines-from-same-history) : Un serveur PEUT exiger que les lignes de base à comparer soient du même historique de ligne de base.

12.7.1 Exemple – rapport DAV:compare-baseline

```
>>REQUEST
REPORT /bl-his/12/bl/14 HTTP/1.1
Host: repo.webdav.com
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:compare-baseline xmlns:D="DAV:">
     <D:href>http://repo.webdav.org/bl-his/12/bl/15</D:href>
   </D:compare-baseline>

>>RESPONSE
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: xxxx

   <?xml version="1.0" encoding="utf-8" ?>
   <D:compare-baseline-report xmlns:D="DAV:">
     <D:added-version>
       <D:href>http://repo.webdav.org/his/23/ver/8</D:href>
     </D:added-version>
     <D:changed-version>
       <D:href>http://repo.webdav.org/his/29/ver/12</D:href>
       <D:href>http://repo.webdav.org/his/29/ver/19</D:href>
     </D:changed-version>
     <D:deleted-version>
       <D:href>http://repo.webdav.org/his/12/ver/4</D:href>
     </D:deleted-version>
   </D:compare-baseline-report>
```

Dans cet exemple, les différences entre baseline 14 et baseline 15 de http://repo.webdav.org/bl-his/12 sont identifiées.

12.8 Sémantique supplémentaire de OPTIONS

Si un serveur prend en charge la caractéristique baseline, il DOIT inclure "baseline" comme champ dans l'en-tête de réponse DAV provenant d'une demande OPTIONS sur toute ressource qui prend en charge des propriétés, rapports, ou méthodes de gestion de version.

12.9 Sémantique supplémentaire de MKCOL

Postcondition supplémentaire : Si un serveur met automatiquement une collection nouvellement créée sous contrôle de ligne de base, toutes les postconditions pour BASELINE-CONTROL s'appliquent au MKCOL.

12.10 Sémantique supplémentaire de COPY

Postcondition supplémentaire : Si la demande crée une nouvelle collection à la destination, et si un serveur met automatiquement une collection nouvellement créée sous contrôle de ligne de base, toutes les postconditions pour BASELINE-CONTROL s'appliquent au COPY.

12.11 Sémantique supplémentaire de CHECKOUT

Précondition supplémentaire : (DAV:must-not-update-baseline-collection) : Si l'URL de demande identifie un membre de la configuration dont la racine est la DAV:baseline-collection d'une ligne de base, la demande DOIT échouer.

12.12 Sémantique supplémentaire de CHECKIN

Préconditions supplémentaires :
(DAV:no-checked-out-baseline-controlled-collection-members) : Si l'URL de demande identifie une configuration de version contrôlée, tous les membres de version contrôlée de la DAV:baseline-controlled-collection de la configuration de version contrôlée DOIVENT être enregistrés.

(DAV:one-version-per-history-per-baseline) : Si l'URL de demande identifie une configuration de version contrôlée, l'ensemble de versions choisi par cette configuration de version contrôlée DOIT contenir au plus une version provenant d'un historique de version, où une version est choisie par une configuration de version contrôlée si la version est identifiée par la propriété DAV:checked-in de tout membre de la configuration qui a sa racine à la collection DAV:baseline-controlled de cette configuration de version contrôlée, ou est identifié par la propriété DAV:checked-in de tout membre de la configuration qui a sa racine à la DAV:baseline-collection de toute sous ligne de base de cette configuration de version contrôlée.

(DAV:cannot-modify-version-controlled-configuration) : Si l'URL de demande identifie un membre de version contrôlée d'une collection de ligne de base contrôlée dont la configuration de version contrôlée est enregistrée, la demande DOIT échouer sauf si la propriété DAV:auto-version de la configuration de version contrôlée désenregistre automatiquement cette configuration de version contrôlée lorsque elle est modifiée.

Postconditions supplémentaires :
(DAV:create-baseline-collection) : Si l'URL de demande identifie une configuration de version contrôlée, la DAV:baseline-collection de la nouvelle ligne de base identifie une collection dont les membres ont le même nom relatif et la même version DAV:checked-in que les membres de la DAV:baseline-controlled-collection de la configuration de version contrôlée au moment de la demande.

(DAV:modify-configuration) : Si l'URL de demande identifie un membre de version contrôlée d'une collection à ligne de base contrôlée, c'est une modification de la configuration de version contrôlée de cette collection à ligne de base contrôlée, et la sémantique standard d'autogestion de version s'applique.

12.13 Sémantique supplémentaire de UPDATE

Préconditions supplémentaires :
(DAV:baseline-controlled-members-must-be-checked-in) : Si l'URL de demande identifie une configuration de version contrôlée, alors tous les membres de version contrôlée de la DAV:baseline-controlled-collection de cette configuration de version contrôlée DOIVENT être enregistrés.

(DAV:must-not-update-baseline-collection) : Si l'URL de demande identifie un membre de la configuration dont la racine est à la DAV:baseline-collection d'une ligne de base, la demande DOIT échouer.

(DAV:cannot-modify-version-controlled-configuration) : Si la demande met à jour la propriété DAV:checked-in de tout membre de version contrôlée d'une collection à ligne de base contrôlée dont la configuration de version contrôlée est enregistrée, la demande DOIT échouer sauf si la propriété DAV:auto-version de la configuration de version contrôlée désenregistre automatiquement cette configuration de version contrôlée lorsque elle est modifiée.

Postconditions supplémentaires :
(DAV:set-baseline-controlled-collection-members) : Si la demande met à jour la propriété DAV:checked-in d'une configuration de version contrôlée, les membres de version contrôlée de la DAV:baseline-controlled-collection de cette configuration de version contrôlée DOIVENT alors avoir été mis à jour afin qu'ils aient le même nom relatif, le même contenu, et les mêmes propriétés mortes que les membres de la DAV:baseline-collection de la ligne de base. En particulier :

- Un membre de version contrôlée pour un certain historique de version DOIT avoir été supprimé si il n'y a pas de membre de version contrôlée pour cet historique de version dans la DAV:baseline-collection de la ligne de base.
- Un membre de version contrôlée pour un certain historique de version DOIT avoir été renommé si son nom relatif à la collection à ligne de base contrôlée est différent de celui du membre de version contrôlée pour cet historique de version dans la DAV:baseline-collection de la ligne de base.
- Un nouveau membre de version contrôlée DOIT avoir été créé pour chaque membre de la DAV:baseline-collection de la ligne de base pour laquelle il n'y a pas de membre de version contrôlée correspondant dans la collection à ligne de base contrôlée.
- Une demande UPDATE DOIT avoir été appliquée à chaque membre de version contrôlée pour un certain historique de version dont la version DAV:checked-in n'est pas la même que celle du membre de version contrôlée pour cet historique de version dans la DAV:baseline-collection de la ligne de base.

(DAV:update-subbaselines) : Si la demande mettait à jour une configuration de version contrôlée dont la DAV:baseline-controlled-collection contenait un membre de ligne de base contrôlée pour une des sous lignes de base de la demande de ligne de base, la propriété DAV:checked-in de la configuration de version contrôlée de ce membre de ligne de base contrôlée DOIT alors avoir été mise à jour pour être cette sous ligne de base. Si la demande mettait à jour une configuration de version contrôlée dont la DAV:baseline-controlled-collection était un membre d'un espace de travail qui contient un membre de ligne de base contrôlée pour une des sous lignes de base de la demande de ligne de base, la propriété DAV:checked-in de la configuration de version contrôlée de ce membre de ligne de base contrôlée DOIT alors avoir été mise à jour pour être cette sous ligne de base.

(DAV:modify-configuration) : Si la demande mettait à jour la propriété DAV:checked-in de tout membre de version contrôlée d'une collection à ligne de base contrôlée, et si cette propriété DAV:checked-in diffère de la propriété DAV:checked-in du membre de version contrôlée correspondant de la DAV:baseline-collection de la ligne de base DAV:checked-in de la DAV:version-controlled-configuration de la collection à ligne de base contrôlée, c'est alors une modification de cette configuration de version contrôlée, et la sémantique standard d'autogestion de version s'applique.

12.14 Sémantique supplémentaire de MERGE

Si la source de fusion est une ligne de base, la cible de fusion est une configuration de version contrôlée pour l'historique de ligne de base de cette ligne de base, où la collection à ligne de base contrôlée de cette configuration de version contrôlée est un membre de la collection identifiée par l'URL de demande.

Préconditions supplémentaires :
(DAV:must-not-update-baseline-collection) : Même sémantique que UPDATE (voir au paragraphe 12.13).

(DAV:cannot-modify-version-controlled-configuration) : Même sémantique que UPDATE (voir au paragraphe 12.13).

Postconditions supplémentaires :
(DAV:merge-baseline) : Si la cible de fusion est une configuration de version contrôlée dont la ligne de base DAV:checked-out n'est pas un descendant de la ligne de base de fusion, celle-ci DOIT avoir été ajoutée au DAV:auto-merge-set d'une configuration de version contrôlée. La version DAV:checked-in de chaque membre de la DAV:baseline-collection de cette ligne de base DOIT avoir été fusionnée dans la DAV:baseline-controlled-collection de cette configuration de version contrôlée.

(DAV:merge-subbaselines) : Si la cible de fusion est une configuration de version contrôlée dont la DAV:baseline-controlled-collection contient un membre de ligne de base contrôlée pour une des sous lignes de base de la ligne de base de fusion, cette sous ligne de base DOIT alors avoir été fusionnées avec la configuration de version contrôlée de ce membre de ligne de base contrôlée. Si la cible de fusion est une configuration de version contrôlée dont la DAV:baseline-controlled-collection est un membre d'un espace de travail qui contient un membre de ligne de base contrôlée pour une des sous lignes de base de la ligne de base de fusion, cette sous ligne de base DOIT alors avoir été fusionnée dans la configuration de version contrôlée de ce membre de ligne de base contrôlée.

(DAV:set-baseline-controlled-collection-members) : Même sémantique que UPDATE (voir au paragraphe 12.13).

(DAV:modify-configuration) : Même sémantique que UPDATE (voir au paragraphe 12.13).

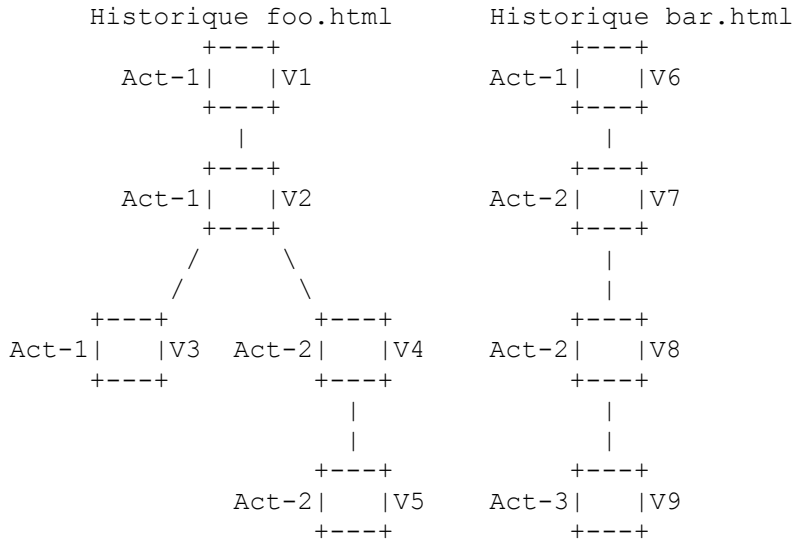
13. Caractéristique Activity

Une activité est une ressource qui choisit un ensemble de versions qui sont sur une seule "ligne de descente", où une ligne de descente est une séquence de versions connectées par une relation de successeur. Si une activité choisit des versions à partir de plusieurs historiques de version, les versions choisies dans chaque historique de version doivent être sur une seule ligne de descente.

Un problème courant qui motive l'utilisation des activités est qu'il est souvent souhaitable d'effectuer plusieurs changements logiques différents dans un seul espace de travail, et de fusionner ensuite de façon sélective un sous ensemble de ces changements logiques avec d'autres espaces de travail. Une activité peut être utilisée pour représenter un seul changement logique lorsque une activité garde trace de toutes les ressources qui ont été modifiées pour effectuer ce seul changement logique. Quand une ressource de version contrôlée est désenregistrée, l'usager spécifie quelle activité devrait être associée à une nouvelle version qui sera créée lorsque cette ressource de version contrôlée sera enregistrée. Il est alors possible de choisir un changement logique particulier pour la fusion dans un autre espace de travail en spécifiant l'activité appropriée dans une demande MERGE.

Un autre problème courant est que bien qu'une ressource de version contrôlée puisse avoir besoin d'avoir plusieurs lignes de descente, tout travail fait par des membres d'une certaine équipe doit être sur une seule ligne de descente (pour éviter de fusionner des membres de l'équipe). Une ressource activity fournit le mécanisme pour régler ce problème. Lorsque une ressource de version contrôlée est désenregistrée, un client peut demander qu'une activité existante soit utilisée ou qu'une nouvelle activité soit créée. La sémantique de Activity assure alors que toutes les versions dans un certain historique de version qui sont associées à une activité soient sur une seule ligne de descente. Si tous les membres d'une équipe partagent une activité commune (ou des sous activités d'une activité commune) alors tous les changements faits par les membres de cette équipe seront sur une seule ligne de descente.

Le diagramme qui suit illustre des activités. La version V5 est la dernière version de foo.html choisie par l'activité Act-2, et la version V8 est la dernière version de bar.html choisie par l'activité Act-2.



Les activités apparaissent sous des noms divers dans les systèmes existants de gestion de version. Lorsque une activité est utilisée pour capturer un changement logique, elle est couramment appelé un "ensemble de changements". Lorsque une activité est utilisée pour capturer une ligne de descente, elle est habituellement appelée une "branche". Lorsque un système accepte les branches et les ensembles de changements, il est souvent utile d'exiger qu'un ensemble de changements particulier se produise sur une branche particulière. Cette relation peut être capturée en faisant de l'activité d'ensemble de changement une "sous activité" de l'activité de branche.

13.1 Propriétés de Activity

Le type de ressource DAV: d'une activité DOIT être DAV:activity.

La caractéristique d'activité introduit les propriétés EXIGÉES suivantes pour une activité.

13.1.1 DAV:activity-version-set (calculé)

Cette propriété identifie chaque version dont la propriété DAV:activity-set identifie cette activité. Plusieurs versions d'un seul historique de version peuvent être choisies par la propriété DAV:activity-version-set d'une activité, mais toutes les versions DAV:activity-version-set provenant d'un certain historique de version doivent être sur une seule ligne de descente provenant de la version racine de cet historique de version.

<!ELEMENT activity-version-set (href*)>

13.1.2 DAV:activity-checkout-set (calculé)

Cette propriété identifie chaque ressource désenregistrée dont le DAV:activity-set identifie cette activité.

<!ELEMENT activity-checkout-set (href*)>

13.1.3 DAV:subactivity-set

Cette propriété identifie chaque activité qui forme une partie du changement logique qui est capturé par cette activité. Une activité se comporte comme si son DAV:activity-version-set était étendu par le DAV:activity-version-set de chaque activité identifiée dans le DAV:subactivity-set. En particulier, les versions dans cet ensemble étendu DOIVENT être sur une seule ligne de descente, et lorsque une activité choisit une version pour une fusion, la dernière version dans cet ensemble étendu est celle qui sera fusionnée.

Un serveur PEUT rejeter les tentatives de modification du DAV:subactivity-set d'une activité.

<!ELEMENT subactivity-set (href*)>

13.1.4 DAV:current-workspace-set (calculé)

Cette propriété identifie chaque espace de travail dont le DAV:current-activity-set identifie cette activité.

<!ELEMENT current-workspace-set (href*)>

13.2 Propriétés supplémentaires de version

La caractéristique activity introduit la propriété EXIGÉE suivante pour une version.

13.2.1 DAV:activity-set

Cette propriété identifie les activités qui déterminent à quels changements logiques cette version contribue, et sur quelles lignes de descente cette version apparaît. Un serveur PEUT restreindre le DAV:activity-set à identifier une seule activité. Un serveur PEUT refuser de permettre la modification de la valeur de la propriété DAV:activity-set d'une version.

<!ELEMENT activity-set (href*)>

13.3 Propriétés supplémentaires de ressource désenregistrée

La caractéristique activity introduit les propriétés EXIGÉES suivantes pour une ressource désenregistrée.

13.3.1 DAV:unreserved

Cette propriété d'une ressource désenregistrée indique si le DAV:activity-set d'une autre ressource désenregistrée associée à l'historique de version de cette ressource de version contrôlée peut avoir une activité qui soit dans la propriété DAV:activity-set de cette ressource désenregistrée.

Un résultat de l'exigence qu'une activité forme une seule ligne de descente à travers un certain historique de version est que si plusieurs ressources désenregistrées pour un certain historique de version sont désenregistrées sans réservation dans une seule activité, seul le premier CHECKIN va réussir. Avant qu'une autre de ces ressources désenregistrées puisse être enregistrée, l'usager va d'abord devoir fusionner dans cette ressource désenregistrée la dernière version choisie par cette activité à partir de cet historique de version, et ensuite modifier le DAV:predecessor-set de cette ressource désenregistrée pour identifier cette version.

```
<!ELEMENT unreserved (#PCDATA)>
  valeur de PCDATA : booléen
```

13.3.2 DAV:activity-set

Cette propriété d'une ressource désenregistrée détermine la propriété DAV:activity-set de la version qui résulte de l'enregistrement de cette ressource.

13.4 Propriétés supplémentaires d'espace de travail

La caractéristique activity introduit la propriété EXIGÉE suivante pour un espace de travail.

13.4.1 DAV:current-activity-set

Cette propriété identifie les activités qui sont en fait exécutées dans cet espace de travail. Lorsque un membre de cet espace de travail est désenregistré, si aucune activité n'est spécifiée dans la demande de désenregistrement, le DAV:current-activity-set sera utilisé. Cela permet à un client ignorant d'une activité de mettre à jour un espace de travail dans lequel le traçage d'activité est exigé. Le DAV:current-activity-set PEUT être restreint à l'identification d'au plus une activité.

```
<!ELEMENT current-activity-set (href*)>
```

13.5 Méthode MKACTIVITY

Une demande MKACTIVITY crée une nouvelle ressource Activity. Un serveur PEUT restreindre la création d'activité à certaines collections, mais un client peut déterminer la localisation de ces collections à partir d'une demande OPTIONS d'un DAV:activity-collection-set.

Surveillance : Si un corps de demande est inclus, il DOIT être un élément XML DAV:mkactivity.

```
<!ELEMENT mkactivity ANY>
```

Si un corps de réponse est inclus pour une demande réussie, il DOIT être un élément XML DAV:mkactivity-response.

```
<!ELEMENT mkactivity-response ANY>
```

La réponse DOIT inclure un en-tête Cache-Control:no-cache.

Préconditions : (DAV:ressource-must-be-null) : Une ressource NE DOIT PAS exister à l'URL de demande.
(DAV:activity-location-ok) : L'URL de demande DOIT identifier une localisation où une activité peut être créée.

Postcondition : (DAV:initialize-activity) : Une nouvelle activité existe à l'URL de demande. Le type de ressource DAV: de l'activité DOIT être DAV:activity.

13.5.1 Exemple - MKACTIVITY

```
>>REQUEST
  MKACTIVITY /act/test-23 HTTP/1.1
  Host: repo.webdav.org
  Content-Length: 0

>>RESPONSE
  HTTP/1.1 201 Created
  Cache-Control: no-cache
```

Dans cet exemple, une nouvelle activité est créée à http://repo.webdav.org/act/test-23.

13.6 Rapport DAV:latest-activity-version

Le rapport DAV:latest-activity-version peut être appliqué à un historique de version pour identifier la dernière version qui est choisie à partir de cet historique de version par une certaine activité.

Surveillance : Le corps de demande DOIT être un élément XML DAV:latest-activity-version.

```
<!ELEMENT latest-activity-version (href)>
```

Le corps de réponse pour une demande réussie DOIT être un élément XML DAV:latest-activity-version-report.

```
<!ELEMENT latest-activity-version-report (href)>
```

Le DAV:href du corps de réponse DOIT identifier la version de l'historique de version qui est membre du DAV:activity-version-set de cette activité et n'avoir pas de descendant qui soit membre du DAV:activity-version-set de cette activité.

Précondition : (DAV:must-be-activity) : Le DAV:href dans le corps de demande DOIT identifier une activité.

13.7 Sémantique supplémentaire de OPTIONS

Si le serveur prend en charge la caractéristique Activity, il DOIT inclure "activity" comme champ dans l'en-tête de réponse DAV à une demande OPTIONS sur toute ressource qui prend en charge des propriétés, rapports, ou méthodes de gestion de version.

Un élément DAV:activity-collection-set PEUT être inclus dans le corps de demande pour identifier des collections qui peuvent contenir des ressources Activity.

Surveillance supplémentaire : Si un corps de demande XML est inclus, il DOIT être un élément XML DAV:options.

```
<!ELEMENT options TOUTE>
  valeur de TOUTE : Séquence d'éléments avec au plus un élément DAV:activity-collection-set.
```

Si un corps de réponse XML pour une demande réussie est inclus, il DOIT être un élément XML DAV:options-response.

```
<!ELEMENT options-response TOUTE>
  valeur de TOUTE : Séquence d'éléments avec au plus un élément DAV:activity-collection-set.
<!ELEMENT activity-collection-set (href*)>
```

Si DAV:activity-collection-set est inclus dans le corps de demande, le corps de réponse pour une demande réussie DOIT contenir un élément DAV:activity-collection-set identifiant les collections qui peuvent contenir des activités. Une collection identifiée PEUT être la collection racine d'une arborescence de collections, dont toutes peuvent contenir des activités. Comme différents serveurs peuvent contrôler des parties différentes de l'espace de noms d'URL, des ressources différentes sur le même hôte PEUVENT avoir des valeurs différentes de DAV:activity-collection-set. Les collections identifiées PEUVENT être situées sur des hôtes différents de ceux de la ressource.

13.8 Sémantique supplémentaire de DELETE

Postcondition supplémentaire : (DAV:delete-activity-reference) : Si une activité est supprimée, toute référence à cette activité dans un DAV:activity-set, DAV:subactivity-set, ou DAV:current-activity-set DOIT être retirée.

13.9 Sémantique supplémentaire de MOVE

Postconditions supplémentaires :
(DAV:update-checked-out-reference) : Si une ressource désenregistrée est déplacée, toute référence à cette ressource dans une propriété DAV:activity-checkout-set DOIT être mise à jour pour se référer à la nouvelle localisation de cette ressource.

(DAV:update-activity-reference) : Si l'URL de demande identifie une activité, toute référence à cette activité dans un DAV:activity-set, DAV:subactivity-set, ou DAV:current-activity-set DOIT être mise à jour pour se référer à la nouvelle localisation de cette activité.

(DAV:update-workspace-reference) : Si l'URL de demande identifie un espace de travail, toute référence à cet espace de travail dans une propriété DAV:current-workspace-set DOIT être mise à jour pour se référer à la nouvelle localisation de cet espace de travail.

13.10 Sémantique supplémentaire de CHECKOUT

Une demande CHECKOUT PEUT spécifier le DAV:activity-set pour la ressource désenregistrée.

Surveillance supplémentaire :

```
<!ELEMENT checkout TOUTE>
  valeur de TOUTE : Séquence d'éléments avec au plus un DAV:activity-set et au plus un DAV:unreserved.
<!ELEMENT activity-set (href+ | new)>
<!ELEMENT new VIDE>
<!ELEMENT unreserved VIDE>
```

Préconditions supplémentaires :
(DAV:one-checkout-per-activity-per-history) : Si une demande d'activité est établie, sauf si DAV:unreserved est spécifié, un autre désenregistrement provenant d'une version de cet historique de version NE DOIT PAS choisir une activité dans cet ensemble d'activités.

(DAV:linear-activity) : Si une demande d'activité est établie, sauf si DAV:unreserved est spécifié, la version choisie DOIT être un descendant de toutes les autres versions de cet historique de version qui a choisi cette activité.

Postconditions supplémentaires :
(DAV:initialize-activity-set) : Le DAV:activity-set de la ressource désenregistrée est réglé comme suit :
- Si DAV:new est spécifié comme DAV:activity-set dans le corps de demande, une nouvelle activité créée par le serveur est utilisée.
- Autrement, si les activités sont spécifiées dans le corps de demande, ces activités sont alors utilisées.
- Autrement, si la ressource de version contrôlée est un membre d'un espace de travail et si le DAV:current-activity-set de l'espace de travail est établi, ces activités sont alors utilisées.
- Autrement, le DAV:activity-set de la version DAV:checked-out est utilisé.

(DAV:initialize-unreserved) : Si DAV:unreserved était spécifié dans le corps de demande, la propriété DAV:unreserved de la ressource désenregistrée DOIT alors être "vrai".

13.10.1 Exemple - CHECKOUT avec une activité

```
>>REQUEST
   CHECKOUT /ws/public/foo.html HTTP/1.1
   Host: www.webdav.org
   Content-Type: text/xml; charset="utf-8"
   Content-Length: xxxx
   <?xml version="1.0" encoding="utf-8" ?>
```

```
   <D:checkout xmlns:D="DAV:">
     <D:activity-set>
       <D:href>http://repo.webdav.org/act/fix-bug-23</D:href>
     </D:activity-set>
   </D:checkout>

>>RESPONSE
   HTTP/1.1 200 OK
   Cache-Control: no-cache
```

Dans cet exemple, le CHECKOUT est effectué dans l'activité http://repo.webdav.org/act/fix-bug-23.

13.11 Sémantique supplémentaire de CHECKIN

Préconditions supplémentaires :
(DAV:linear-activity) : Toute version qui est dans l'historique de version de la ressource désenregistrée et dont le DAV:activity-set identifie une activité à partir du DAV:activity-set de la ressource désenregistrée DOIT être un ancêtre de la ressource désenregistrée.

(DAV:atomic-activity-checkin) : Si l'URL de demande identifie une activité, le serveur PEUT faire échouer la demande si une des ressources désenregistrées dans le DAV:activity-checkout-set de cette activité ou d'une sous activité de cette activité ne peut pas être enregistrée.

Postconditions supplémentaires :
(DAV:initialize-activity-set) : Le DAV:activity-set de la nouvelle version DOIT avoir été initialisé comme le même que le DAV:activity-set de la ressource désenregistrée.

(DAV:activity-checkin) : Si l'URL de demande identifiait une activité, le serveur DOIT avoir réussi à appliquer la demande CHECKIN à chaque ressource désenregistrée dans le DAV:activity-checkout-set de cette activité et de toute sous activité de cette activité.

13.12 Sémantique supplémentaire de MERGE

Si l'élément DAV:source du corps de demande identifie une activité, pour chaque historique de version contenant une version choisie par cette activité, la dernière version choisie par cette activité est alors une source de fusion. Noter que les versions choisies par une activité sont l'union des versions dans son DAV:activity-version-set et des versions choisies par les activités dans son DAV:subactivity-set.

Surveillance supplémentaire : <!ELEMENT checkin-activity VIDE>

Postcondition supplémentaire : (DAV:checkin-activity) : Si DAV:checkin-activity est spécifié dans le corps de demande, et si l'élément DAV:source dans le corps de demande identifie une activité, une demande CHECKIN DOIT avoir été bien appliquée à cette activité avant que les sources de fusion aient été déterminées.

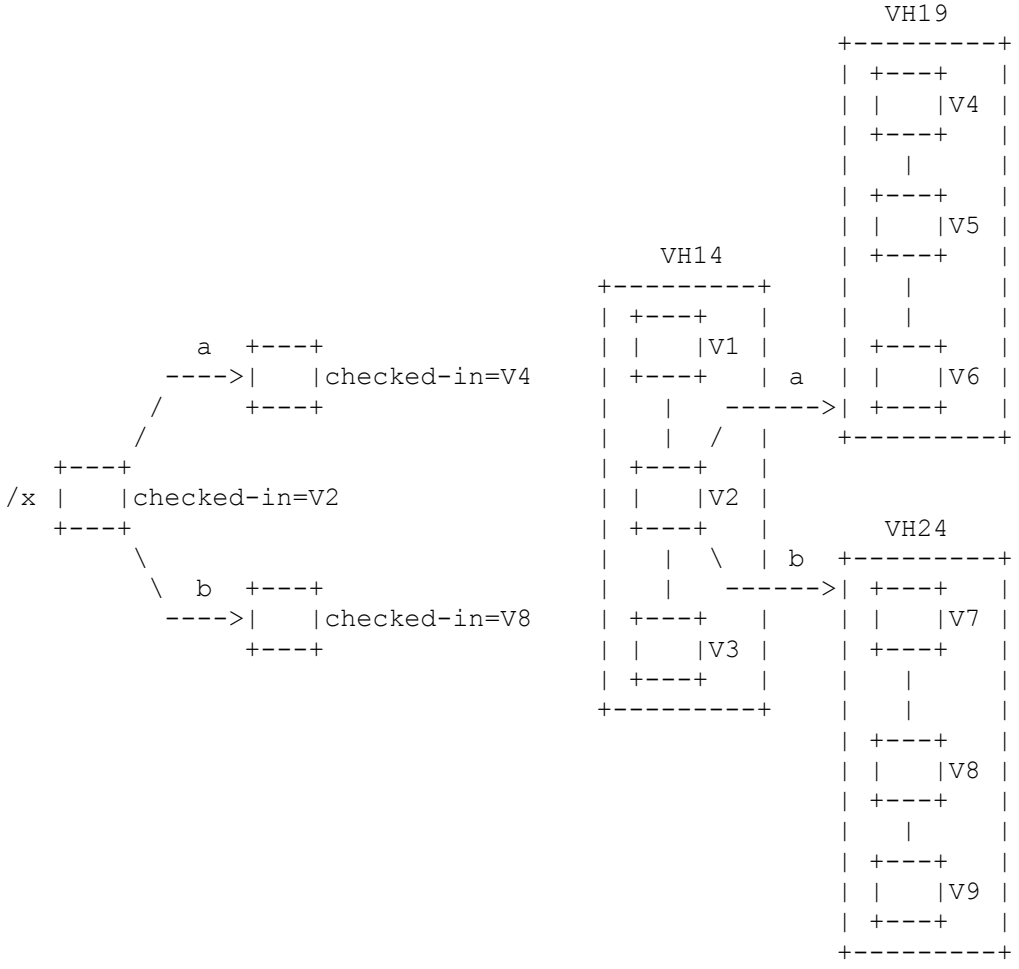
14. Caractéristique Version-Controlled-Collection

Comme avec toute ressource versionisable, lorsque une collection est mise sous contrôle de version, une ressource d'historique de version est créée pour contenir les versions pour cette collection de versions contrôlées. Afin de préserver la sémantique standard de gestion de versions (une version d'une collection ne devrait pas être modifiable) une version de collection enregistre seulement des informations sur les liens de version contrôlée de cette collection.

Afin de séparer proprement une modification de l'espace de noms d'une modification du contenu ou des propriétés mortes, une version d'une collection n'a pas de membre, mais enregistre plutôt dans sa propriété DAV:version-controlled-binding-set le nom du lien et de la ressource d'historique de version de chaque membre interne de version contrôlée de cette collection. Si autrement, une version de collection contenait des liens à d'autres versions, la création d'une nouvelle version d'une ressource exigerait de créer une nouvelle version de toutes les versions de collection qui contiennent cette ressource, ce qui causerait l'enchevêtrement des activités. Par exemple, supposons qu'une activité "feature-12" ait créé une nouvelle version de /x/y/a.html. Si une version de collection contenait des liens avec les versions de ses membres, une nouvelle

version de /x/y devrait être créée pour contenir la nouvelle version de /x/y/a.html, et une nouvelle version de /x devrait être créée pour contenir la nouvelle version de /x/y. Supposons maintenant qu'une activité "bugfix-47" ait créé une nouvelle version de /x/z/b.html. Là encore, une nouvelle version de /x/z et une nouvelle version de /x doit être créée pour contenir la nouvelle version de /x/y/b.html. Mais il est maintenant impossible de fusionner juste "bugfix-47" dans un autre espace de travail sans "feature-12", parce que la version de /x qui contient la version désirée de /x/z/b.html contient aussi la version de /x/y/a.html créée pour "feature-12". Si à la place une version de collection enregistre juste le nom du lien et de la ressource d'historique de version de chaque membre interne de version contrôlée, changer la version choisie par un membre de cette collection n'exigera pas une nouvelle version de la collection. La nouvelle version est toujours dans le même historique de version de sorte qu'aucune nouvelle version de collection n'est requise, et "feature-12" et "bugfix-47" ne seront pas entremêlés.

Dans l'exemple qui suit, il y a trois historiques de version, VH14, VH19, et VH24, où VH14 contient des versions d'une collection. La collection de versions contrôlées /x a la version V2 de l'historique de version VH14 comme version DAV:checked-in. Comme V2 a enregistré deux liens de version contrôlée, un avec le nom de lien "a" à l'historique de version VH19, et l'autre avec le nom de lien "b" à l'historique de version VH24, /x DOIT avoir deux liens de version contrôlée, l'un nommé "a" à une ressource de version contrôlée pour l'historique VH19, et l'autre nommé "b" à une ressource de version contrôlée pour l'historique VH24. La ressource de version contrôlée /x/a a actuellement V4 de VH19 comme sa version de DAV:checked-in, tandis que /x/b a V8 de VH24 comme sa version de DAV:checked-in.



Pour toute demande (par exemple, DELETE, MOVE, COPY) qui modifie un lien de version contrôlée d'une collection enregistrée de versions contrôlées, la demande DOIT échouer sauf si la collection de versions contrôlées a une propriété DAV:auto-version qui va automatiquement désenregistrer la collection de versions contrôlées lorsque elle est modifiée.

Bien qu'une version de collection n'enregistre que les liens de version contrôlée d'une collection, une collection de versions contrôlées PEUT contenir les deux liens de version contrôlée et de version non contrôlée. Les liens de version non contrôlée ne sont pas sous contrôle de version, et peuvent donc être ajoutés ou supprimés sans désenregistrer la collection de versions contrôlées.

Noter qu'une version de collection ne capture qu'un sous ensemble défini de l'état d'une collection. En particulier, une version d'une collection capture ses propriétés mortes et ses liens aux ressources de version contrôlée, mais pas ses

propriétés vives ou ses liens aux ressources de version non contrôlée.

Lorsqu un serveur prend en charge la caractéristique working-ressource, un client peut désenregistrer une version de collection pour créer une collection de travail. À la différence d'une collection de versions contrôlées, qui contient des liens avec des ressources de version contrôlée et des ressources de version non contrôlée, une collection de travail contient des liens avec des ressources d'historique de version et des ressources de version non contrôlée. En particulier, une collection de travail est initialisée à contenir des liens avec les ressources d'historique de version spécifiées par le DAV:version-controlled-binding-set de la version de collection désenregistrée. Les membres d'une collection de travail peuvent alors être supprimés ou déplacés dans une autre collection de travail. Les ressources de version non contrôlée peuvent être ajoutées à une collection de travail par des méthodes telles que PUT, COPY, et MKCOL. Lorsque une collection de travail est enregistrée, une demande VERSION-CONTROL est automatiquement appliquée à chaque membre de version non contrôlée de la collection de travail, et chaque membre de version non contrôlée est remplacé par son historique de version nouvellement créé. Le DAV:version-controlled-binding-set de la nouvelle version résultant de l'enregistrement dans une collection de travail contient le nom du lien de l'URL d'historique de version pour chaque membre de la collection de travail.

14.1 Propriétés de collection version contrôlée

Une collection de versions contrôlées a toutes les propriétés d'une collection et d'une ressource de version contrôlée. En plus, la caractéristique collection de version contrôlée introduit la propriété EXIGÉE suivante pour une collection de versions contrôlées.

14.1.1 DAV:eclipsed-set (calculé)

Cette propriété identifie les membres internes de version non contrôlée de la collection qui éclipsent actuellement un membre interne de version contrôlée de la collection.

```
<!ELEMENT eclipsed-set (binding-name*)>
<!ELEMENT binding-name (#PCDATA)>
  valeur de PCDATA : segment d'URL
```

Une demande UPDATE ou MERGE peut donner une collection de versions contrôlées à un membre interne de version contrôlée qui a le même nom qu'un membre interne de version non contrôlée existant. Dans ce cas, le membre interne de version non contrôlée prend la préséance et est dit "éclipser" le nouveau membre interne de version contrôlée. Si le membre interne de version non contrôlée est retiré (par exemple, par un DELETE ou un MOVE) le membre interne de version contrôlée est exposé.

14.2 Propriétés de version de collection

Une version de collection a toutes les propriétés d'une version. De plus, la caractéristique version-controlled-collection introduit la propriété EXIGÉE suivante pour une version de collection.

14.2.1 DAV:version-controlled-binding-set (protégé)

Cette propriété capture le nom et l'historique de version de chaque membre interne de version contrôlée d'une collection.

```
<!ELEMENT version-controlled-binding-set (version-controlled-binding*)>
<!ELEMENT version-controlled-binding (binding-name, version-history)>
<!ELEMENT binding-name (#PCDATA)>
  valeur de PCDATA : segment d'URL
<!ELEMENT version-history (href)>
```

14.3 Sémantique supplémentaire de OPTIONS

Si le serveur prend en charge la caractéristique version-controlled-collection, il DOIT inclure un champ "version-controlled-collection" dans l'en-tête de réponse DAV à une demande OPTIONS sur toute ressource qui prend en charge des propriétés, rapports, ou méthodes de gestion de version.

14.4 Sémantique supplémentaire de DELETE

Précondition supplémentaire : (DAV:cannot-modify-checked-in-parent) : Si l'URL de demande identifie une ressource de version contrôlée, le DELETE DOIT échouer lorsque la collection contenant la ressource de version contrôlée est une collection de versions contrôlées enregistrée, sauf si la sémantique de DAV:auto-version va automatiquement désenregistrer la collection de versions contrôlées.

14.5 Sémantique supplémentaire de MKCOL

Précondition supplémentaire : Si la demande crée une nouvelle ressource qui est automatiquement placée sous contrôle de version, toutes les préconditions pour VERSION-CONTROL s'appliquent à la demande.

Postcondition supplémentaire : Si la nouvelle collection est mise automatiquement sous contrôle de version, toutes les postconditions pour VERSION-CONTROL s'appliquent à la demande.

14.6 Sémantique supplémentaire de COPY

Précondition supplémentaire : (DAV:cannot-copy-collection-version) : Si la source de la demande est une version de collection, la demande DOIT échouer.

14.7 Sémantique supplémentaire de MOVE

Préconditions supplémentaires :
(DAV:cannot-modify-checked-in-parent) : Si la source de la demande est une ressource de version contrôlée, la demande DOIT échouer lorsque la collection qui contient la source est une collection de versions contrôlées enregistrée, sauf si la sémantique de DAV:auto-version désenregistre automatiquement cette collection de versions contrôlées.

(DAV:cannot-modify-destination-checked-in-parent) : Si la source de la demande est une ressource de version contrôlée, la demande DOIT échouer lorsque la collection contenant la destination est une collection de versions contrôlées enregistrée, sauf si la sémantique de DAV:auto-version désenregistre automatiquement cette collection de versions contrôlées.

14.8 Sémantique supplémentaire de VERSION-CONTROL

Précondition supplémentaire : (DAV:cannot-modify-checked-in-parent) : Si le parent de l'URL de demande est une collection de versions contrôlées enregistrée, la demande DOIT échouer sauf si la sémantique de DAV:auto-version désenregistre automatiquement cette collection de versions contrôlées.

Postcondition supplémentaire : (DAV:new-version-controlled-collection) : Si le corps de demande identifiait un version de collection, la collection à l'URL de demande DOIT contenir un membre interne de version contrôlée pour chaque lien DAV:version-controlled-binding spécifié dans le DAV:version-controlled-binding-set de la version de collection, où le nom et le DAV:version-history du membre interne DOIVENT être le DAV:binding-name et le DAV:version-history spécifiés par le DAV:version-controlled-binding. Si le membre interne est un membre d'un espace de travail, et si il y a un autre membre de l'espace de travail pour le même historique de version, ces deux membres DOIVENT identifier la même ressource de version contrôlée ; autrement, une demande VERSION-CONTROL avec une version choisie par le serveur de l'historique de version DOIT avoir été appliquée à l'URL pour ce membre interne.

14.9 Sémantique supplémentaire de CHECKOUT

Postcondition supplémentaire : (DAV:initialize-version-history-bindings) : Si la demande a été appliquée à une version de collection, la nouvelle collection de travail DOIT être initialisée à contenir un lien sur chaque ressource d'historique identifiée dans le DAV:version-controlled-binding-set de cette version de collection.

14.10 Sémantique supplémentaire de CHECKIN

Postconditions supplémentaires :
(DAV:initialize-version-controlled-bindings) : Si l'URL de demande identifiait une collection de versions contrôlées, le DAV:version-controlled-binding-set de la nouvelle version de collection DOIT alors contenir un DAV:version-controlled-binding qui identifie le nom du lien et l'historique de version pour chaque lien de version contrôlée de la collection de version contrôlée.

(DAV:version-control-working-collection-members) : Si l'URL de demande identifiait une collection de travail, une demande VERSION-CONTROL DOIT avoir été automatiquement appliquée à chaque membre de version non contrôlée de la collection de travail, et chaque membre de version non contrôlée DOIT avoir été remplacé par son historique de version nouvellement créé. Si un membre de collection de travail n'était pas une collection de versions non contrôlées, chaque membre de la collection de versions non contrôlées DOIT avoir été placé sous contrôle de version avant que la collection de versions non contrôlées ait été placée sous contrôle de version. Le DAV:version-controlled-binding-set de la nouvelle version de collection DOIT contenir un DAV:version-controlled-binding qui identifie le nom du lien et l'URL d'historique de version pour chaque membre de la collection de travail.

14.11 Sémantique supplémentaire de UPDATE et de MERGE

Postconditions supplémentaires :
(DAV:update-version-controlled-collection-members) : Si la demande modifiait la version DAV:checked-in d'une collection de versions contrôlées, les membres de version contrôlée de cette collection de versions contrôlées DOIVENT alors avoir été mis à jour. En particulier :

- Un membre interne de version contrôlée DOIT avoir été supprimé si son historique de version n'est pas identifié par le DAV:version-controlled-binding-set de la nouvelle version DAV:checked-in.
- Un membre interne de version contrôlée pour un certain historique de version DOIT avoir été renommé si son nom de lien diffère du DAV:binding-name pour cet historique de version dans le DAV:version-controlled-binding-set de la nouvelle version DAV:checked-in.
- Un nouveau membre interne de version contrôlée DOIT avoir été créé lorsque un historique de version est identifié par le DAV:version-controlled-binding-set de la version DAV:checked-in, mais qu'il n'y avait pas de membre de la collection de versions contrôlées pour cet historique de version. Si un nouveau membre de version contrôlée est dans un espace de travail qui a déjà une ressource de version contrôlée pour cet historique de version, le nouveau membre de version contrôlée DOIT alors être juste un lien (c'est-à-dire, un autre nom) pour cette ressource de version contrôlée existante. Autrement, le contenu et les propriétés mortes du nouveau membre de version contrôlée DOIVENT avoir été initialisés comme étant ceux de la version spécifiée pour cet historique de version par la demande. Si aucune version n'est spécifiée pour cet historique de version par la demande, la version choisie est définie par le serveur.

15. Considérations d'internationalisation

La présente spécification a été conçue pour être conforme à la politique de l'IETF sur les jeux de caractères et les langages [RFC2277]. Précisément, lorsque des chaînes lisibles par l'homme existent dans le protocole, soit leur jeu de caractères est déclaré explicitement, soit des mécanismes XML sont utilisés pour spécifier le jeu de caractères utilisé. De plus, ces chaînes lisibles par l'homme ont toutes la capacité d'exprimer le langage naturel de la chaîne.

La plupart des chaînes lisibles par l'homme apparaissent dans le présent protocole dans des propriétés, telles que DAV:creator-displayname. Comme défini dans la [RFC2518], les propriétés ont leurs valeurs données comme XML. XML a des dispositions explicites pour l'étiquetage et le codage de jeux de caractères, et exige que les processeurs XML lisent les éléments XML codés, au minimum, en utilisant le codage UTF-8 [RFC2279] du plan multilingue ISO 10646. Le paramètre charset de l'en-tête Content-Type, avec l'attribut XML "encoding", fournissent les informations d'identification de jeu de caractères pour les processeurs MIME et XML. L'utilisation appropriée de l'en-tête charset avec XML est décrite dans la [RFC3023]. XML fournit aussi une capacité d'étiquetage de langage pour spécifier le langage du contenu d'un élément XML particulier. XML utilise soit les étiquettes enregistrées par l'IANA (voir la [RFC3066]) soit les étiquettes de

langage de [ISO639] dans l'attribut "xml:lang" d'un élément XML pour identifier le langage de son contenu et des ses attributs.

Les applications DeltaV, construites par dessus WebDAV, sont soumises aux exigences d'internationalisation spécifiées dans la RFC 2518, Section 16. En bref, ces exigences rendent obligatoire l'utilisation de l'étiquetage du jeu de caractères XML, le codage du jeu de caractères, et les capacités d'étiquetage de langage. De plus, elles recommandent fortement de lire les instructions de la RFC 3023 sur l'utilisation des types de supports MIME pour le transport XML et l'utilisation de l'en-tête charset.

Au sein de la présente spécification, une étiquette est une chaîne lisible par l'homme qui est insérée dans l'en-tête Label et comme XML dans les corps d'entité de demande. Lorsque elle est utilisée dans l'en-tête Label, la valeur de l'étiquette est à échappement d'URL et est codée en utilisant UTF-8.

16. Considérations pour la sécurité

Toutes les considérations pour la sécurité de WebDAV discutées dans la Section 17 de la RFC 2518 s'appliquent aussi à la gestion de version WebDAV. Certains aspects du protocole de gestion de version aident à traiter les risques pour la sécurité introduits pas WebDAV, mais d'autres aspects peuvent augmenter ces risques pour la sécurité. Ces questions sont détaillées ci-dessous.

16.1 Audit et traçabilité

WebDAV augmente la facilité avec laquelle un client distant peut modifier des ressources sur un site de la Toile, mais cela augmente aussi le risque que des informations importantes soient écrasées et perdues, soit par une erreur de l'utilisateur, soit par malveillance. L'utilisation de la gestion de version WebDAV peut aider à régler ce problème en garantissant que les informations précédentes sont sauvegardées sous la forme de versions immuables, et qui sont donc facilement disponibles pour restitution ou restauration. De plus, l'historique de version fournit un enregistrement du moment où les changements ont été faits, et par qui. Lorsque les demandes sont authentifiées de façon appropriée, le mécanisme d'historique fournit un journal d'audit clair pour les changements apportés aux ressources de la Toile. Cela peut souvent améliorer significativement la capacité à identifier la source du problème de sécurité, et par là aider à s'en garder à l'avenir.

16.2 Besoin accru de contrôle d'accès

La gestion de version WebDAV fournit diverses liaisons entre les éléments d'information en relation. Cela peut augmenter le risque que des erreurs d'authentification ou d'autorisation permettent à un client de localiser des informations sensibles. Par exemple, si l'historique de version n'est pas protégé de façon appropriée par le contrôle d'accès, un client peut utiliser l'historique de version d'une ressource publique pour identifier les versions ultérieures de cette ressource alors que l'usager avait l'intention de les garder confidentielles. Cela augmente le besoin d'une authentification fiable et d'une autorisation précise.

Un client de gestion de version WebDAV devrait être conçu pour traiter un mélange de réponses 200 (OK) et 403 (Interdit) aux tentatives d'accès aux propriétés et rapports qui sont acceptés par une ressource. Par exemple, un usager particulier peut être autorisé à accéder au contenu et aux propriétés mortes d'une ressource de version contrôlée, mais n'être pas autorisé à accéder aux propriétés DAV:checked-in, DAV:checked-out, ou DAV:version-history de cette ressource.

16.3 Sécurité par l'obscurité

Bien qu'on reconnaisse que "l'obscurité" n'est pas un moyen efficace de sécurité, elle est souvent une bonne technique pour faire que les honnêtes gens restent honnêtes. Au sein du présent protocole, les URL de version, les URL d'historique de version, et les URL de ressource de travail sont générés par le serveur et peuvent être proprement obscurcis pour ne pas attirer l'attention sur eux. Par exemple, une version de "http://foobar.com/reviews/salaries.html" peut recevoir un URL tel que "http://foobar.com/repo/4934943".

16.4 Déni de service

Le mécanisme d'autogestion de version fourni par WebDAV peut résulter en la création d'un grand nombre de ressources sur le serveur, car chaque mise à jour d'une ressource peut résulter en la création d'une nouvelle ressource de version. Cela augmente le risque d'attaque de déni de service qui épuise les capacités de mémorisation d'un serveur. Ce risque est particulièrement significatif parce qu'il peut être un résultat non intentionnel de quelque chose comme un dispositif d'auto sauvegarde agressif fourni par un client d'édition. Un serveur peut réduire ce risque en utilisant les techniques de mémorisation de deltas pour minimiser le coût des versions supplémentaires, et en limitant l'autogestion de version aux client verrouillés, diminuant par là le nombre de créations de versions involontaires.

17. Considérations en rapport avec l'IANA

Le présent document utilise l'espace de noms défini par la RFC 2518 pour les éléments XML. Toutes les autres considérations relatives à l'IANA tirées de la RFC 2518 sont aussi applicables à la gestion de version WebDAV.

18. Propriété intellectuelle

L'IETF ne prend pas position sur la validité et la portée de tout droit de propriété intellectuelle ou autres droits qui pourrait être revendiqués au titre de la mise en œuvre ou l'utilisation de la technologie décrite dans le présent document ou sur la mesure dans laquelle toute licence sur de tels droits pourrait être ou n'être pas disponible ; pas plus qu'elle ne prétend avoir accompli aucun effort pour identifier de tels droits. Les informations sur les procédures de l'ISOC au sujet des droits dans les documents de l'ISOC figurent dans les BCP 78 et BCP 79.

Des copies des dépôts d'IPR faites au secrétariat de l'IETF et toutes assurances de disponibilité de licences, ou le résultat de tentatives faites pour obtenir une licence ou permission générale d'utilisation de tels droits de propriété par ceux qui mettent en œuvre ou utilisent la présente spécification peuvent être obtenues sur répertoire en ligne des IPR de l'IETF à http://www.ietf.org/ipr.

L'IETF invite toute partie intéressée à porter son attention sur tous copyrights, licences ou applications de licence, ou autres droits de propriété qui pourraient couvrir les technologies qui peuvent être nécessaires pour mettre en œuvre la présente norme. Prière d'adresser les informations à l'IETF à ietf-ipr@ietf.org.

19. Remerciements

Ce protocole est le résultat de la collaboration des auteurs et du reste de l'équipe de conception de DeltaV : Boris Bokowski, Bruce Cragun (Novell), Jim Doubek (Macromedia), David Durand (INSO), Lisa Dusseault (Xythos), Chuck Fay (FileNet), Yaron Goland, Mark Hale (Interwoven), Henry Harbury (Merant), James Hunt, Jeff McAffer (OTI), Peter Raymond (Merant), Juergen Reuter, Edgar Schwarz (Marconi), Eric Sedlar (Oracle), Bradley Sergeant, Greg Stein, et John Vasta (Rational). Nous tenons à remercier la fondation établie pour nous par les auteurs des protocoles WebDAV et HTTP sur lesquels ce protocole s'appuie, et les précieux retours des groupes de travail WebDAV et DeltaV.

20. Références

[ISO639] ISO, "Code for the representation of names of languages", ISO 639:1988, 1998.

[RFC2026] S. Bradner, "Le processus de normalisation de l'Internet -- Révision 3", (BCP0009) octobre 1996. *(Remplace* RFC1602, RFC1871) (*MàJ par* RFC3667, RFC3668, RFC3932, RFC3979, RFC3978, RFC5378, RFC6410)

[RFC2119] S. Bradner, "Mots clés à utiliser dans les RFC pour indiquer les niveaux d'exigence", BCP 14, mars 1997.

[RFC2227] J. Mogul, P. Leach, "Simple mesure du nombre d'accès et limitation d'usage pour HTTP", octobre 1997. *(P.S.)*

[RFC2279] F. Yergeau, "UTF-8, un format de transformation de la norme ISO 10646", janvier 1998. (*Obsolète, voir* RFC3629) *(D.S.)*

[RFC2396] T. Berners-Lee, R. Fielding et L. Masinter, "Identifiants de ressource uniformes (URI) : Syntaxe générique",

août 1998. (*Obsolète, voir* RFC3986)

[RFC2518] Y. Goland, E. Whitehead, A. Faizi, S. Carter et D. Jensen, "Extensions HTTP pour la création répartie -- WEBDAV", février 1999. *(Obsolète, voir la* RFC4918*)*

[RFC2616] R. Fielding et autres, "Protocole de transfert hypertexte -- HTTP/1.1", juin 1999. *(D.S., MàJ par* 2817, 6585*)*

[RFC3023] M. Murata, S. St.Laurent et D. Kohn, "Types de support XML", janvier 2001. (*Obsolète, voir* RFC7303)

[RFC3066] H. Alvestrand, "Étiquettes pour l'identification des langues", BCP 47, janvier 2001. (*Obs., voir* RFC4646)

Appendice A Classification des ressources

Le présent document introduit plusieurs sortes différentes de ressources de gestion de version, comme les ressources de version contrôlée, les versions, les ressources désenregistrées, et les ressources d'historique de version. Lorsque des clients découvrent des ressources sur un serveur, ils peuvent trouver utile de classer ces ressources (par exemple, pour prendre des décisions d'UI sur le choix des options d'icône et de menu).

Les clients devraient classer une ressource en examinant les valeurs des propriétés DAV:supported-method-set (voir au paragraphe 3.1.3) et DAV:supported-live-property-set (voir au paragraphe 3.1.4) de cette ressource.

La liste qui suit montre les propriétés vives et les méthodes prises en charge pour chaque sorte de ressource de gestion de version. Lorsque une caractéristique facultative introduit une nouvelle sorte de ressource de gestion de version, cette caractéristique est notée entre parenthèses à la suite du nom de cette sorte de ressource de gestion de version. Si une propriété vive ou méthode est facultative pour une sorte de ressource de version, la caractéristique qui introduit cette propriété vive ou méthode est notée entre parenthèses à la suite du nom de la propriété vive ou méthode.

A.1 URL non transposé conforme à DeltaV (URL qui n'identifie aucune ressource)

Méthodes prises en charge :
- PUT [RFC2616]
- MKCOL [RFC2518]
- MKACTIVITY (activité)
- VERSION-CONTROL (espace de travail)
- MKWORKSPACE (espace de travail)

A.2 Ressource conforme à DeltaV

Propriétés vives prises en charge :
- DAV:comment
- DAV:creator-display-name
- DAV:supported-method-set
- DAV:supported-live-property-set
- DAV:supported-report-set
- toutes les propriétés définies dans WebDAV [RFC2518].

Méthodes prises en charge :
- REPORT
- toutes les méthodes définies dans WebDAV [RFC2518]
- toutes les méthodes définies dans HTTP/1.1 [RFC2616].

A.3 Collection conforme à DeltaV

Propriétés vives prises en charge :
- toutes les propriétés de ressource conformes à DeltaV.

Méthodes prises en charge :

- BASELINE-CONTROL (ligne de base)
- toutes les méthodes de ressource conformes à DeltaV.

A.4 Ressource versionisable

Propriétés vives prises en charge :
- DAV:workspace (espace de travail)
- DAV:version-controlled-configuration (ligne de base)
- toutes les propriétés de ressource conformes à DeltaV.

Méthodes prises en charge :
- VERSION-CONTROL
- toutes les méthodes de ressource conformes à DeltaV.

A.5 Ressource de version contrôlée

Propriétés vives prises en charge :
- DAV:auto-version
- DAV:version-history (historique de version)
- DAV:workspace (espace de travail)
- DAV:version-controlled-configuration (ligne de base)
- toutes les propriétés de ressource conformes à DeltaV.

Méthodes prises en charge :
- VERSION-CONTROL
- MERGE (fusion)
- toutes les méthodes de ressource conformes à DeltaV.

A.6 Version

Propriétés vives prises en charge :
- DAV:predecessor-set
- DAV:successor-set
- DAV:checkout-set
- DAV:version-name
- DAV:checkout-fork (in-place-checkout ou ressource de travail)
- DAV:checkin-fork (in-place-checkout ou ressource de travail)
- DAV:version-history (historique de version)
- DAV:label-name-set (étiquette)
- DAV:activity-set (activité)
- toutes les propriétés de ressource conformes à DeltaV..

Méthodes prises en charge :
- LABEL (étiquette)
- CHECKOUT (ressource de travail)
- toutes les méthodes de ressource conformes à DeltaV.

A.7 Ressource de version contrôlée enregistrée

Propriétés vives prises en charge :
- DAV:checked-in
- toutes propriétés de ressource de version contrôlée.

Méthodes prises en charge :
- CHECKOUT (checkout-in-place)
- UPDATE (mise à jour)
- toutes les méthodes de ressource de version contrôlée.

A.8 Ressource désenregistrée

Propriétés vives prises en charge :

- DAV:checked-out
- DAV:predecessor-set
- DAV:checkout-fork (in-place-checkout ou ressource de travail)
- DAV:checkin-fork (in-place-checkout ou ressource de travail)
- DAV:merge-set (fusion)
- DAV:auto-merge-set (fusion)
- DAV:unreserved (activité)
- DAV:activity-set (activité)

Méthodes prises en charge :
- CHECKIN (checkout-in-place ou ressource de travail)
- toutes les méthodes de ressource conformes à DeltaV.

A.9 Ressource de version contrôlée désenregistrée (checkout-in-place)

Propriétés vives prises en charge :
- toutes propriétés de ressource de version contrôlée.
- toutes propriétés de ressource désenregistrée.

Méthodes prises en charge :
- UNCHECKOUT
- toutes méthodes de ressource de version contrôlée.
- toutes méthodes de ressource désenregistrée.

A.10 Ressource de travail (working-resource)

Propriétés vives prises en charge :
- toutes propriétés de ressource conformes à DeltaV
- toutes propriétés de ressource désenregistrée
- DAV:auto-update.

Méthodes prises en charge :
- toutes méthodes de ressource désenregistrée.

A.11 Historique de version (version-history)

Propriétés vives prises en charge :
- DAV:version-set
- DAV:root-version
- toutes propriétés de ressource conformes à DeltaV.

Méthodes prises en charge :
- toutes méthodes de ressource conformes à DeltaV.

A.12 Workspace (espace de travail)

Propriétés vives prises en charge :
- DAV:workspace-checkout-set
- DAV:baseline-controlled-collection-set (ligne de base)
- DAV:current-activity-set (activité)
- toutes propriétés de collection conformes à DeltaV.

Méthodes prises en charge :
- toutes les méthodes de collection conformes à DeltaV.

A.13 Activity (activité)

Propriétés vives prises en charge :
- DAV:activity-version-set
- DAV:activity-checkout-set
- DAV:subactivity-set
- DAV:current-workspace-set
- toutes propriétés de ressource conformes à DeltaV.

Méthodes prises en charge :
- toutes les méthodes de ressource conformes à DeltaV.

A.14 Collection de version contrôlée (version-controlled-collection)

Propriétés vives prises en charge :
- DAV:eclipsed-set
- toutes propriétés de ressource de version contrôlée.

Méthodes prises en charge :
- toutes méthodes de ressource de version contrôlée.

A.15 Collection de versions contrôlées (version-controlled-collection)

Propriétés vives prises en charge :
- DAV:version-controlled-binding-set
- toutes les propriétés de version.

Méthodes prises en charge :
- toutes les méthodes de version.

A.16 Configuration de version contrôlée (ligne de base)

Propriétés vives prises en charge :
- DAV:baseline-controlled-collection
- toutes propriétés de ressource de version contrôlée.

Méthodes prises en charge :
- toutes méthodes de ressource de version contrôlée.

A.17 Ligne de base (baseline)

Propriétés vives prises en charge :
- DAV:baseline-collection
- DAV:subbaseline-set
- toutes propriétés de version.

Méthodes prises en charge :
- toutes méthodes de version.

A.18 Configuration de version contrôlée désenregistrée (ligne de base)

Propriétés vives prises en charge :
- DAV:subbaseline-set
- toutes propriétés de configuration de version contrôlée.

Méthodes prises en charge :
- toutes méthodes de configuration de version contrôlée.

Adresse des auteurs

Geoffrey Clemm
Rational Software
20 Maguire Road, Lexington, MA 02421
mél : geoffrey.clemm@rational.com

Jim Amsden
IBM
3039 Cornwallis, Research Triangle Park, NC 27709
mél : jamsden@us.ibm.com

Tim Ellison
IBM
Hursley Park, Winchester,
UK S021 2JN
mél : tim_ellison@uk.ibm.com

Christopher Kaler
Microsoft
One Microsoft Way, Redmond,
WA 90852
mél : ckaler@microsoft.com

Jim Whitehead
UC Santa Cruz,Computer Science dpt
1156 High Street, Santa Cruz,
CA 95064
mél : ejw@cse.ucsc.edu

Déclaration complète de droits de reproduction

Remerciement
Le financement de la fonction d’éditeur des RFC est actuellement fourni par la Internet Society.